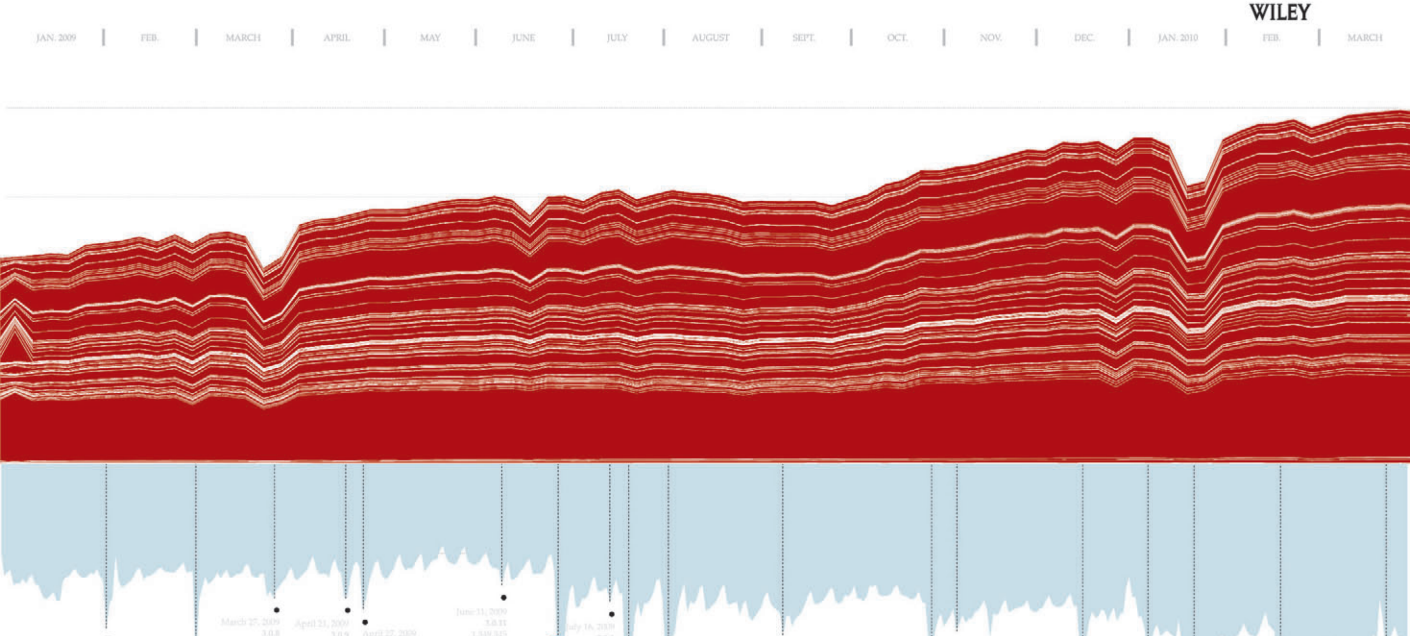
图灵程序设计丛书

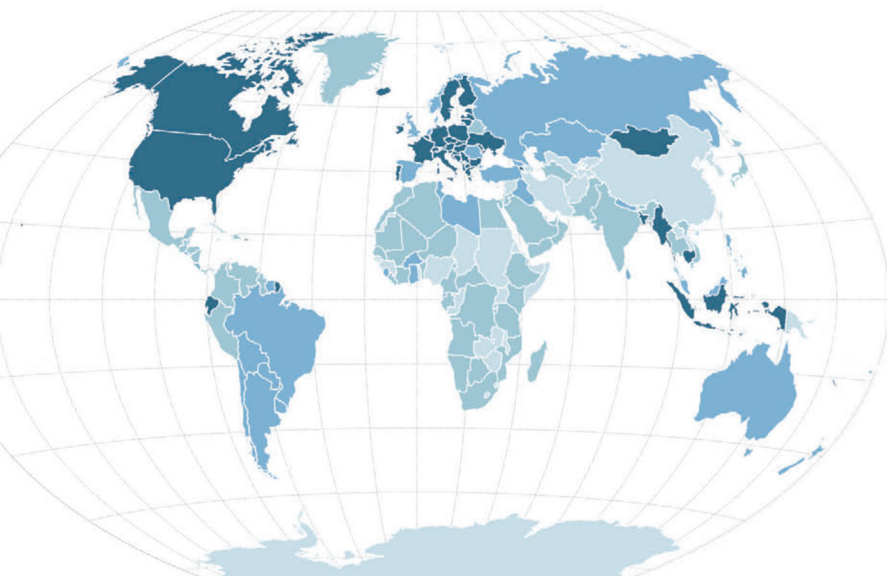


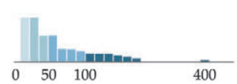
鲜活的数据

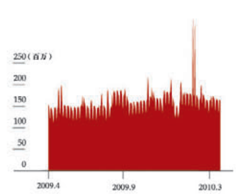
数据可视化指南

[美] Nathan Yau 著

向怡宁 译







Sources: Mozilla, Internet World Stats; Design by: Nathan Yau, FlowingData

人民邮电出版社
POSTS & TELECOM PRESS

“本书就像是一封写给Python、R、地图和数据的情书。”

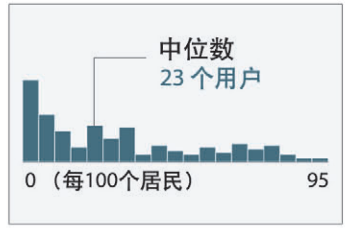
——FlowingData读者评论

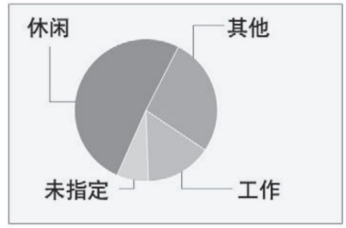
“我是Nathan Yan的博客FlowingData的忠实粉丝，本书还没出来我就预订了。果然，它完全符合我的预期：各种各样的分析、数据资源和绝对精美的图表。”

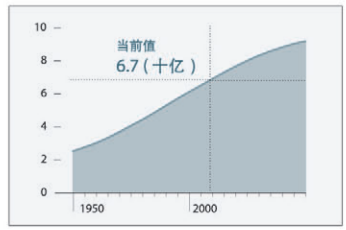
——亚马逊读者评论

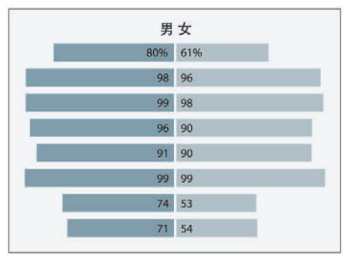
“本书写得很好，思路清晰，实例丰富，如果你经常与数据打交道，选择本书错不了。”

——亚马逊读者评论









在生活中，数据几乎无处不在，任我们取用。然而，同样的数据给人的感觉可能会千差万别：或冰冷枯燥，让人望而生畏、百思不解其意；或生动有趣，让人一目了然、豁然开朗。为了达到后一种效果，我们需要采用一种特别的方式来展示数据，来解释、分析和应用它。这就是数据可视化技术。

Nathan Yau是这一创新领域的先锋。在本书中，他根据数据可视化的工作流程，先后介绍了如何获取数据，将数据格式化，用可视化工具（如R）生成图表，以及在图形编辑软件（如Illustrator）中修改以使图表达到最佳效果。本书介绍了数十种方法（如柱形图、饼图、折线图和散点图等），以创造性的视觉方式生动讲述了有关数据的故事。翻开本书，思维之门会豁然大开，你会发现有那么多的手段去赋予数据全新的意义！

本书主要内容包括：

- 学习如何用视觉化表示方式来呈现数据，让读者看到不一样的信息；
- 发现数据背后的故事；
- 探索不同的数据来源，确定有效的展示格式；
- 试验并对比不同的可视化工具；
- 寻找数据中的趋势和模式，并以适当的图表来展现它们；
- 设定明确的目标，并用其指引你的可视化过程。

WILEY
www.wiley.com

图灵社区：www.ituring.com.cn
新浪微博：@图灵教育 @图灵社区
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议 计算机/数据可视化

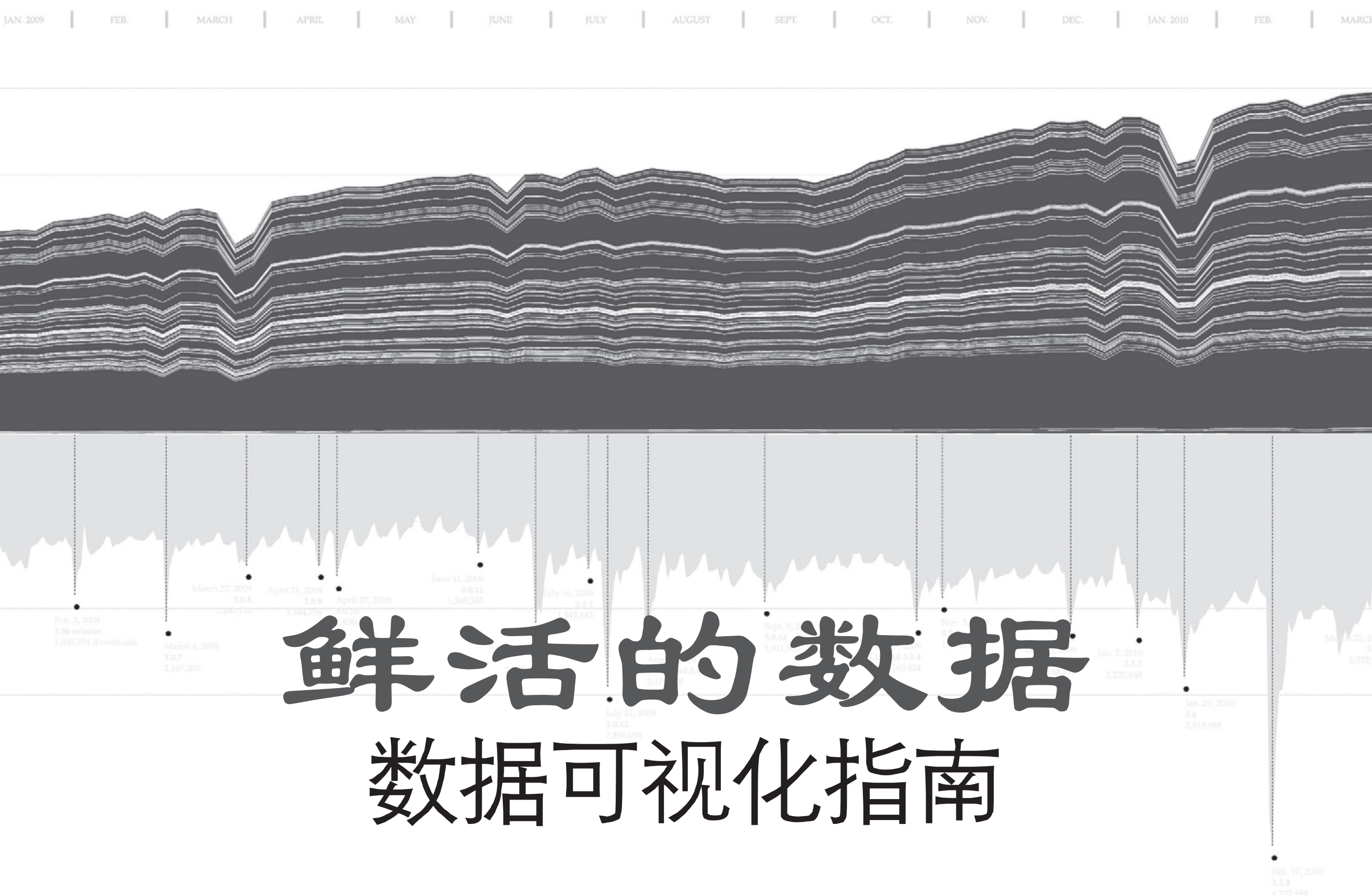
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-29381-7

9 787115 293817 >

ISBN 978-7-115-29381-7
定价：69.00元

TURING 图灵程序设计丛书



鲜活的数据

数据可视化指南

[美] Nathan Yau 著
向怡宁 译

人 民 邮 电 出 版 社
北 京

图书在版编目（C I P）数据

鲜活的数据 : 数据可视化指南 / (美) 邱 (Yau,N.)
著 ; 向怡宁译. -- 北京 : 人民邮电出版社, 2012.10
(图灵程序设计丛书)
书名原文: Visualize This : The FlowingData
Guide to Design，Visualization，and Statistics
ISBN 978-7-115-29381-7

Ⅰ. ①鲜… Ⅱ. ①邱… ②向… Ⅲ. ①数据收集－指
南 Ⅳ. ①TP311.13-62

中国版本图书馆CIP数据核字(2012)第220656号

内 容 提 要

本书是一本系统介绍数据可视化的图书。书中主要阐述了如何将冰冷枯燥的数据转换成易于理解、生动有趣、主题清晰的图表。作者根据数据可视化的一般顺序，先后介绍了如何获取数据，将数据格式化，然后用可视化工具（如 R）生成图表，最后在图形处理软件（如 Illustrator）中修改完善，使图表达到最佳的可视化效果。本书详细介绍了柱形图、饼图、折线图和散点图等图表的绘制方法及各自的优缺点，还用专门的一章介绍与地图相关的数据可视化技巧。

本书实例丰富、图文并茂，适合数据分析师、视觉设计师和对数据感兴趣的开发人员学习提高。

图灵程序设计丛书
鲜活的数据：数据可视化指南

◆ 著 [美] Nathan Yau
译 向怡宁
责任编辑 朱 巍
执行编辑 罗词亮
◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京 印刷
◆ 开本：800×1000 1/16
印张：18.5 彩插 16
字数：437千字 2012年 9 月第 1 版
印数：1－4 000册 2012年 9 月北京第 1 次印刷
著作权合同登记号 图字：01-2012-3283号
ISBN 978-7-115-29381-7

定价：69.00元
读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版权声明

引　言

数据不是什么新鲜玩意。早在几个世纪之前，人们就开始对数据进行量化分析并为之绘制表格了。然而在为FlowingData（我创建的一个有关设计、可视化和统计的网站）写作时，我发觉这一领域在过去数年间出现了爆炸式的发展，而且未来还会更加蓬勃。科技的进步使得收集和存储数据变得轻而易举，而互联网则让我们摆脱了时间和空间的束缚。如果运用得当，这种数据的“财富”能够提供丰富的信息，帮助人们更明智地制定决策、更清楚地传达理念，而且能让我们以更为客观的角度去审视自己对世界和自身的看法。

随着2009年年中Data.gov网站的上线，美国政府数据公开化进程发生了一次重大转变。这是一套综合的数据目录系统，由各级联邦政府机构提供，表现出各组织及官方的透明度和责任感。比如说，国民有权利了解政府把税收收入都花在了哪里，而在此之前美国政府给人的感觉就像一个黑箱。Data.gov上的很多数据其实在许多网站中都能找到，但现在它们都被会聚在一起，而且有着统一的格式，更加便于人们进行分析和可视化。除了Data.gov之外，联合国也有类似的网站UNdata，英国很快也发布了Data.gov.uk，而像纽约、旧金山和伦敦等全球许多城市也都参与到了数据公开这一潮流中来。

如今的网站也变得越来越开放，有数千个API（应用编程接口）在鼓励和“怂恿”着开发人员去调用网站已有的数据做各种事情。比如Twitter和Flickr就提供了覆盖面极广的API，开发人员可以自由定制与网站本身完全不同、五花八门的用户界面。API编目网站ProgrammableWeb目前已收录超过2000个API①。诸如Infochimps和Factual这样的应用最近也大量涌现出来，它们存在的目的就是向人们提供结构化的数据。

在个人层面，我们可以在Facebook上结交朋友，在Foursquare上分享所在的位置，或者在Twitter上发布自己的最新动态，这所有的一切都只需要点击几次鼠标或者敲击几下键盘就能实现。一些针对性更强的应用则方便我们记录品尝过什么美食、体重几何、情绪高低等林林总总的事情。几乎可以这样说，只要你想对自己的某个方面进行追踪，就会有这样一款应用来帮助你实现愿望。

数据就静静地待在我们生活的每一个角落。园子里已经果实累累，正等待着我们去采摘。对大多数人来说，真正有意思的并不是数据本身，而是数据背后蕴涵的信息。人们都希望知道他们的数据有何意义，而如果你能帮助他们，那么你就会大受欢迎。难怪Google首席经济学家Hal

① 截至2012年5月底，该网站已收录6100多个API。——译者注（如无特殊说明，下文中所有脚注均为译者注）

Varian会说统计学家将是未来十年内最迷人的职业，而这绝不仅仅是因为统计学家长得好看（尽管以极客们的别样眼光来看，我们确实长得让人赏心悦目）。

可视化

要想探索和理解那些大型的数据集，可视化是最有效的途径之一。把数字置于视觉空间中，我们的大脑或者读者的大脑就会更容易发现其中潜藏的模式。人类对图形的理解能力确实很强，往往能够从中发现一些通过常规统计方法很难挖掘到的信息。

John Tukey是我最喜爱的统计学家，也是探索性数据分析理论（Exploratory Data Analysis）的缔造者。他精通各种统计方法和工具，而且深信图形技术在其中亦占有一席之地。他坚信，图形的呈现方式会让人们得到许多出乎意料的结果。只需对数据进行可视化，我们就能从中发现大量信息，而且很多情况下这也是我们制定明智决策或描述复杂事件所需要的唯一手段。

比如说，在2009年美国的失业率遭遇了一次大幅增长。2007年的全美平均失业率是4.6%，2008年上涨到了5.8%。而到了2009年9月，突然就攀升至9.8%。但是这些全国平均数字只揭示了事件的一部分，它们只是概括了整个国家的总体状况。有哪些地区的失业率高于其他地区？又有哪些地区并未受到很大波及？我们无法从中获得答案。

图0-1用一系列美国地图更为完整地说明了情况，而且我们只需略扫一眼就能回答上面的问题。颜色较深的县失业率相对较高，而颜色较浅的县失业率较低。在2009年的地图上（图0-2），我们可以看到美国西部和东部大多数地区的失业率都超过了10%，而中西部地区则未受到太大影响。



图0-1　2004—2009年美国失业率分布图

如果手上只有单纯的电子表格，要想找到其中蕴涵的地区性或周期性的模式就会很花时间，而只靠前面那些全国平均数字则完全不可能。而用地图呈现之后，虽然增加了许多县的数据，但读者的理解程度反而提高了。这些地图有可能帮助当局决定往哪些地区划拨救济金或提供其他形式的援助。

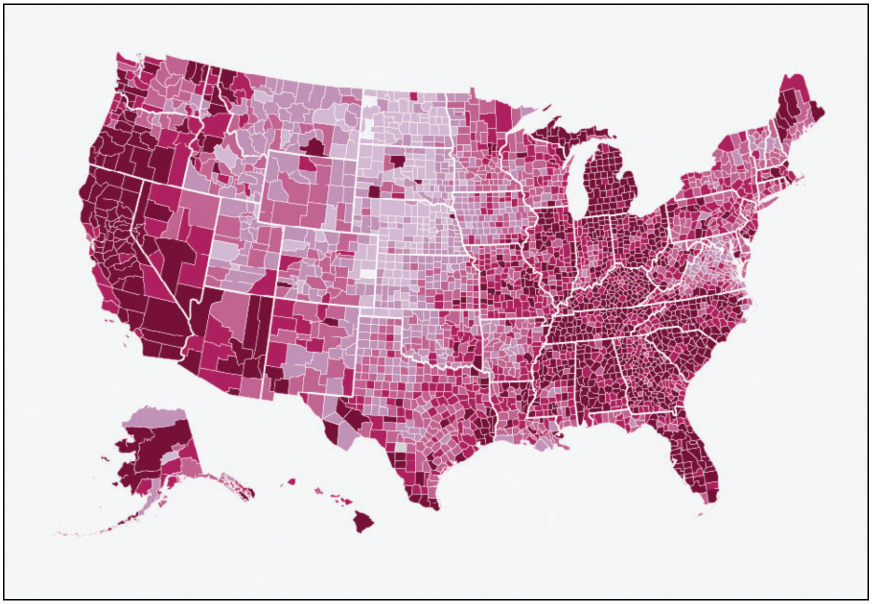


图0-2 2009年失业率分布图

这个例子的绝妙之处在于，用于产生地图的数据都是免费的，由美国劳工统计局直接面向公众开放。尽管找到这些数据并不是那么轻而易举，但它们确实就在某个地方听候我们的差遣，而且还有更多格式化的数据正等待着我们作更好的视觉处理。

比如说，《美国统计摘要》（*The Statistical Abstract of the United States*）[①]就含有数百个数据表格（见图0-3），但没有任何图表。这简直是天赐的良机，我们可以在此基础上进行加工，展现整个国家的概貌。这个过程将会非常有趣。不久前我用图形描绘了其中的部分表格（见图0-4），很快就得到了美国近年来结婚率及离婚率、邮政资费、用电量等信息的直观变化情况。单纯的表格形式很难阅读，读者只能得到一些零散的数值，而在图表化视图中，人们能够轻易地发现变化的趋势和模式，而且一眼就能作出比较。

①《美国统计摘要》由美国商务部人口普查局整理发布，是美国社会、政治和经济机构的标准统计摘要。每一篇都提供文字说明并在表格下方注明出处。自1878年首次出版以来，每年出版一次。因经费问题，在2012年出版最后一期（第131期）后该刊即停刊。——编者注

表126：1990—2007年各州结婚、离婚数字及比率

[**2,443.5 represents 2,443,500.** By place of occurence. See Appendix III]

State	Marriages [1]						Divorces [3]					
	Number (1,000)			Rate per 1,000 population [2]			Number (1,000)			Rate per 1,000 population [2]		
	1990	2000	2007	1990	2000	2007	1990	2000	2007	1990	2000	2007
U.S. [4]	**2,443.5**	**2,329.0**	**2,204.6**	**9.8**	**8.3**	**7.3**	**1,182.0**	**(NA)**	**(NA)**	**4.7**	**4.1**	**3.6**
Alabama	43.1	45.0	42.4	10.6	10.3	9.2	25.3	23.5	19.8	6.1	5.4	4.3
Alaska [5]	5.7	5.6	5.8	10.2	8.9	8.4	2.9	2.7	3.0	5.5	4.4	4.3
Arizona [5]	36.8	38.7	39.5	10.0	7.9	6.2	25.1	21.6	24.5	6.9	4.4	3.9
Arkansas	36.0	41.1	33.7	15.3	16.0	11.9	16.8	17.9	16.8	6.9	6.9	5.9
California	237.1	196.9	225.8	7.9	5.9	6.2	128.0	(NA)	(NA)	4.3	(NA)	(NA)
Colorado	32.4	35.6	29.2	9.8	8.6	6.0	18.4	(NA)	21.2	5.5	(NA)	4.4
Connecticut	26.0	19.4	17.3	7.9	5.9	4.9	10.3	6.5	10.7	3.2	2.0	3.1
Delaware	5.6	5.1	4.7	8.4	6.7	5.5	3.0	3.2	3.9	4.4	4.2	4.5
District of Columbia	5.0	2.8	2.1	8.2	5.4	3.6	2.7	1.5	1.0	4.5	3.0	1.6
Florida	141.8	141.9	157.6	10.9	9.3	8.6	81.7	81.9	86.4	6.3	5.3	4.7
Georgia	66.8	56.0	64.0	10.3	7.1	6.7	35.7	30.7	(NA)	5.5	3.9	(NA)
Hawaii	18.3	25.0	27.3	16.4	21.2	21.3	5.2	4.6	(NA)	4.6	3.9	(NA)
Idaho	14.1	14.0	15.4	13.9	11.0	10.3	6.6	6.9	7.4	6.5	5.4	4.9
Illinois	100.6	85.5	75.3	8.8	7.0	5.9	44.3	39.1	32.8	3.8	3.2	2.6
Indiana	53.2	34.5	51.2	9.6	5.8	8.1	(NA)	(NA)	(NA)	(NA)	(NA)	(NA)
Iowa	24.9	20.3	20.1	9.0	7.0	6.7	11.1	9.4	7.8	3.9	3.3	2.6
Kansas	22.7	22.2	18.6	9.2	8.3	6.7	12.6	10.6	9.2	5.0	4.0	3.3
Kentucky	49.8	39.7	33.6	13.5	10.0	7.9	21.8	21.6	19.7	5.8	5.4	4.6
Louisiana	40.4	40.5	32.8	9.6	9.3	7.6	(NA)	(NA)	(NA)	(NA)	(NA)	(NA)
Maine	11.9	10.5	10.1	9.7	8.3	7.7	5.3	5.8	5.9	4.3	4.6	4.5
Maryland	46.3	40.0	35.5	9.7	7.7	6.3	16.1	17.0	17.4	3.4	3.3	3.1
Massachusetts	47.7	37.0	38.4	7.9	6.0	6.0	16.8	18.6	14.5	2.8	3.0	2.2
Michigan	76.1	66.4	59.1	8.2	6.7	5.9	40.2	39.4	35.5	4.3	4.0	3.5
Minnesota	33.7	33.4	29.8	7.7	6.9	5.7	15.4	14.8	(NA)	3.5	3.1	(NA)
Mississippi	24.3	19.7	15.7	9.4	7.1	5.4	14.4	14.4	14.2	5.5	5.2	4.9
Missouri	49.1	43.7	39.4	9.6	7.9	6.7	26.4	26.5	22.4	5.1	4.8	3.8
Montana	6.9	6.6	7.1	8.6	7.4	7.4	4.1	2.1	3.6	5.1	2.4	3.7
Nebraska	12.6	13.0	12.4	8.0	7.8	7.0	6.5	6.4	5.5	4.0	3.8	3.1
Nevada	120.6	144.3	126.4	99.0	76.7	49.3	13.3	18.1	16.6	11.4	9.6	6.5
New Hampshire	10.5	11.6	9.4	9.5	9.5	7.1	5.3	7.1	5.1	4.7	5.8	3.9
New Jersey	58.7	50.4	45.4	7.6	6.1	5.2	23.6	25.6	25.7	3.0	3.1	3.0
New Mexico [5]	13.3	14.5	11.2	8.8	8.3	5.7	7.7	9.2	8.4	4.9	5.3	4.3
New York [5]	154.8	162.0	130.6	8.6	8.9	6.8	57.9	62.8	55.9	3.2	3.4	2.9
North Carolina	51.9	65.6	68.1	7.8	8.5	7.5	34.0	36.9	37.4	5.1	4.8	4.1
North Dakota	4.8	4.6	4.2	7.5	7.3	6.6	2.3	2.0	1.5	3.6	3.2	2.4
Ohio	98.1	88.5	70.9	9.0	7.9	6.2	51.0	49.3	37.9	4.7	4.4	3.3
Oklahoma	33.2	15.6	26.2	10.6	4.6	7.3	24.9	12.4	18.8	7.7	3.7	5.2
Oregon	25.3	26.0	29.4	8.9	7.8	7.8	15.9	16.7	14.8	5.5	5.0	4.0
Pennsylvania	84.9	73.2	71.1	7.1	6.1	5.7	40.1	37.9	35.3	3.3	3.2	2.8
Rhode Island	8.1	8.0	6.8	8.1	8.0	6.4	3.8	3.1	3.0	3.7	3.1	2.8
South Carolina	55.8	42.7	31.4	15.9	10.9	7.1	16.1	14.4	14.4	4.5	3.7	3.3
South Dakota	7.7	7.1	6.2	11.1	9.6	7.7	2.6	2.7	2.4	3.7	3.6	3.1
Tennessee	68.0	88.2	65.6	13.9	15.9	10.6	32.3	33.8	29.9	6.5	6.1	4.9
Texas	178.6	196.4	179.9	10.5	9.6	7.5	94.0	85.2	79.5	5.5	4.2	3.3
Utah	19.4	24.1	22.6	11.2	11.1	8.6	8.8	9.7	8.9	5.1	4.5	3.4
Vermont	6.1	6.1	5.3	10.9	10.2	8.6	2.6	5.1	2.4	4.5	8.6	3.8
Virginia	71.0	62.4	58.0	11.4	9.0	7.5	27.3	30.2	29.5	4.4	4.3	3.8
Washington	46.6	40.9	41.8	9.5	7.0	6.5	28.8	27.2	28.9	5.9	4.7	4.5
West Virginia	13.0	15.7	13.0	7.2	8.7	7.2	9.7	9.3	9.0	5.3	5.2	5.0
Wisconsin	38.9	36.1	32.2	7.9	6.8	5.8	17.8	17.6	16.1	3.6	3.3	2.9
Wyoming	4.9	4.9	4.8	10.7	10.3	9.3	3.1	2.8	2.9	6.6	5.9	5.5

NA Not available. [1] Data are counts of marriages performed, except as noted. [2] Based on total population residing in area; population enumerated as of April 1 for 1990 and 2000; estimated as of July 1 for all other years. [3] Includes annulments. [4] U.S. total for the number of divorces is an estimate which includes states not reporting. Beginning 2000, divorce rates based solely on the combined counts and populations for reporting states and the District of Columbia. The collection of detailed data of marriages and divorces was suspended in January 1996. [5] Some figures for marriages are marriage licenses issued.

Source: U.S. National Center for Health Statistics, National Vital Statistics Reports (NVSR), *Births, Marriages, Divorces, and Deaths: Provisional Data for 2007*, Vol. 56, No. 21, July 14, 2008 and prior reports.

图0-3　美国统计摘要网站中的表格

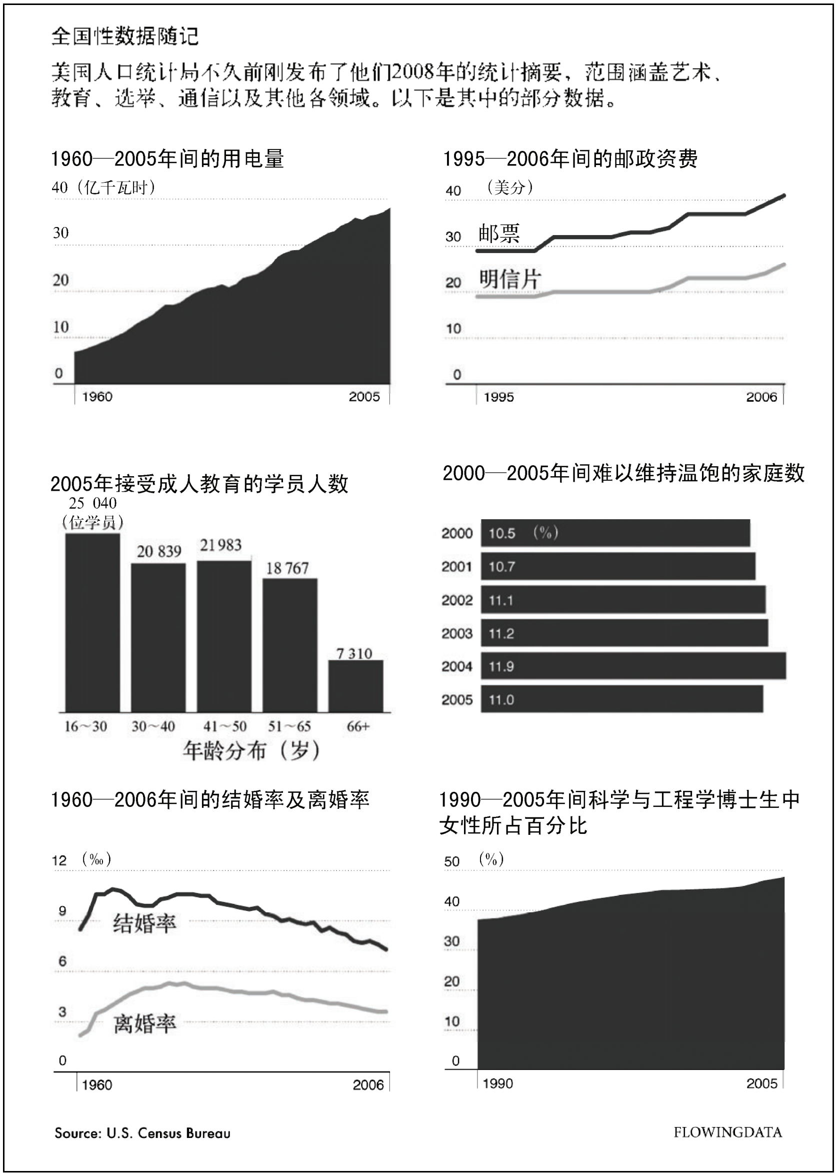


图0-4　美国统计摘要网站数据的图表化视图

类似《纽约时报》、《华盛顿邮报》这样的新闻机构很擅长让数据变得栩栩如生、易于理解。它们对已有数据的利用也许是最充分的，因为经常会有相关主题的新闻故事见诸报端。有时故事中还会插入数据图表以强调不同的观点，而有时只需要图表就能讲述整个故事。

在传统媒体向网络媒体转型的过程中，图形的应用变得更加普及。如今的新闻机构中都已设立了专门处理交互、图表或地图数据的各种部门，比如《纽约时报》就专门为“计算机辅助报道”成立了一个新闻编辑部，旗下的记者都专注于用数据来报道新闻。而《纽约时报》的图形编辑部处理起大量数据来也同样得心应手。

即使是在流行文化领域，可视化也占据了自己的一席之地。Stamen Design是一家以在线交互闻名的可视化公司，他们在过去数年中一直都在对每年的MTV音乐录影带大奖颁奖时期的Twitter状态进行追踪。Stamen Design每一次的设计都与之前有所不同，但其核心一直保持不变：实时展现人们在Twitter上的热门话题。2009年Kanye West在Taylor Swift发表获奖感言时突然暴走[①]，我们通过Stamen Design的追踪可以很容易地了解人们对他这种行径的看法。

现在看来，我们发现这个领域中也有偏重情绪而非分析的一面，对可视化的定义开始变得模糊起来。在很长一段时间内人们都认为，可视化就是关于量化后的事实：我们把它们作为工具来识别事物发展的模式，转而为分析研究提供帮助。但可视化并不仅仅与冰冷的事实有关。就如同Stamen Design的追踪设计一样，它有着很强的娱乐因素，为观众提供了另一种方式去关注颁奖典礼，并在过程中与其他粉丝进行互动。Jonathan Harris的设计也是一个很好的例子。在他的*We Feel Fine*（我们感觉良好）[②]和*Whale Hunt*（捕鲸）[③]等作品中，Harris并不是出于分析角度，而是围绕着故事本身来进行设计，而且这些故事以人类情感为中心，超越了单纯的数字和分析行为。

图表和图形逐渐也超出了工具的范畴，发展为传达理念的载体。GraphJam和Indexed之类的网站[④]就喜欢运用文氏图[⑤]、饼图等形式来戏谑流行歌曲及文化，用红白黑等颜色组合来讥讽政客，或者谴责虐待动物的行为。我自己也在这个方向上作了一些尝试，在FlowingData上发表了系列漫画Data Underload（数据低负荷）。在图0-5中，我用图形表现了美国电影协会评选出的一些经典电影台词——非常无厘头，但很有趣（至少对我来说如此）。[⑥]

▶访问http://datafl.ws/underload，欣赏FlowingData网站上的更多Data Underload漫画。

① Kanye West是美国黑人说唱歌手，他在2009年的MTV音乐录像带颁奖典礼上冲上台打断了当届最佳录影带大奖得主Taylor的获奖感言，并声称该奖项应属于Beyonce。

② 作品见http://www.wefeelfine.org/。本书第1章有详细介绍。

③ 作品见http://thewhalehunt.org/。设计师将在一次为期7天的捕鲸行动中所拍摄的大量照片用可视化方式进行排列。读者可以选择马赛克、时间线或轮辐等浏览模式。

④ 这两个网站可以被归类于“每日一漫画”类型的网站，只不过通常以简单的图表形式来展现笑料。

⑤ 文氏图（Venn diagram）是一种常见的用于表示集合或类的粗略草图。例如用圆圈A和圆圈B分别代表两个集合，那么两个圆圈交叠的区域就是集合A与集合B的交集。

⑥ 原图内并未给出相关台词的原文，考虑到国内读者可能不太熟悉出处，特在图中添加。

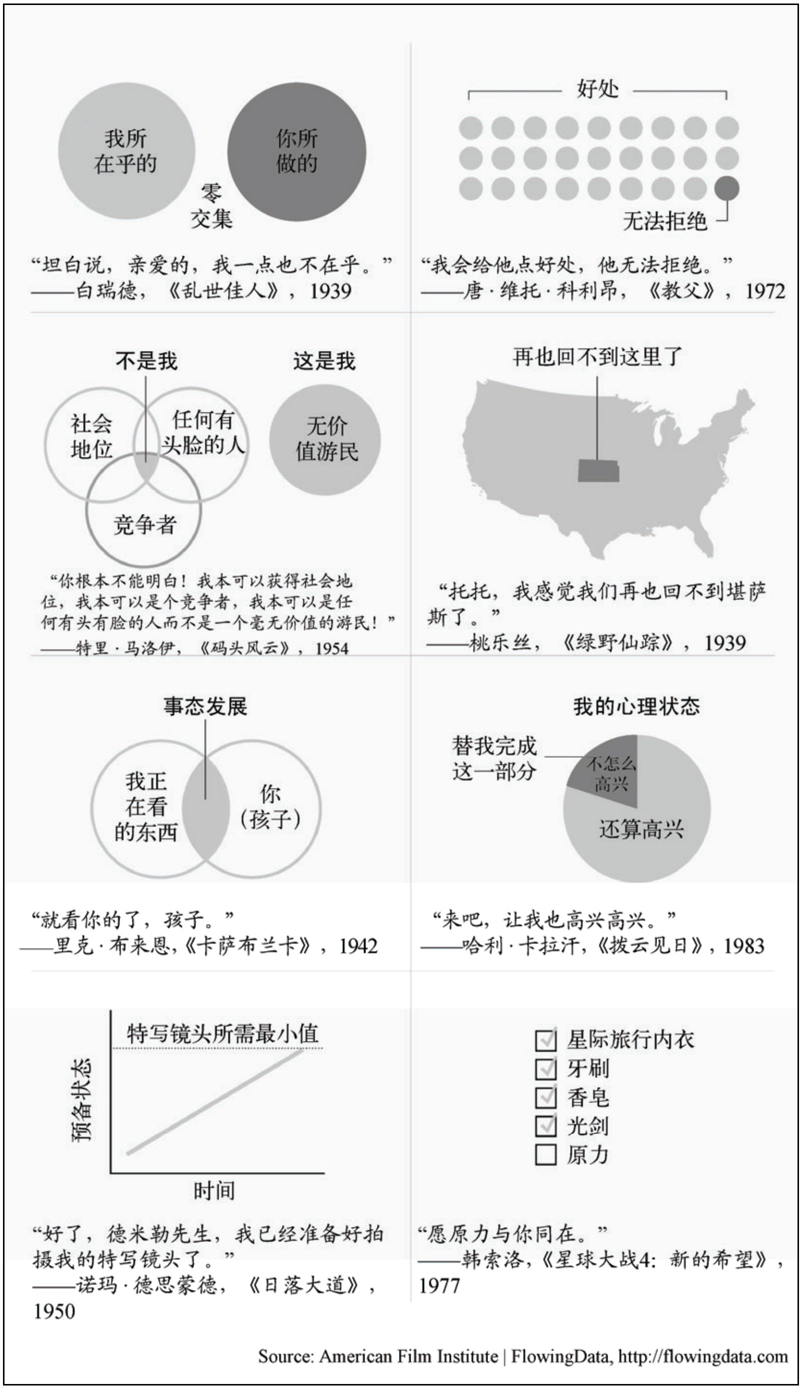


图0-5　图表形式的电影台词

那么，到底什么是可视化呢？每个人都有自己的答案。有些人认为只有严格意义上的传统图形图表才是可视化。而另一些人的观点则更加开放，他们认为只要是在表现数据，不管是数据艺术品还是微软Excel表格，都可以算是可视化。我个人较为倾向于后者，但有时也发现自己站在前一阵营。毕竟，这一问题上孰是孰非并不是那么重要，只要能达成我们的目的就行了。

不管可视化是什么，我们绘制演示用的图例也好、进行数据分析也好、用数据来报道新闻也罢，最终其实都是在寻求真相。在某些时候，统计也会产生错误的假象，但造成错觉的并不是数字本身，而是运用数字的人。有时候这是有意为之，但更多情况下是疏忽大意所致。如果我们不知道如何创建合适的图形，或者不知道如何客观地看待数据，那么就会产生谬误。但只要我们掌握了适当的可视化技巧和处理方式，就能更加自信地陈述观点，并且对自己的发现感觉良好。

学习数据

我在大学一年级时开始接触统计学，当时它是一门必修的基础课，但与我的专业电气工程并没有太大关系。讲课的教授热情极高，而且对这一领域乐此不疲。他上课时喜欢在教室的台阶上来回走动，身体语言极为丰富，而且不时鼓励身边的学生参与讨论。我从未遇到过如此兴奋的老师，而且毫无疑问，正是这种精神吸引我进入了数据领域，最终在四年后考上了统计学的研究生。

在本科四年中，统计学就是数据分析、频率分布和假设检验，而我一直乐在其中。我觉得观察数据集，探索其中的趋势、模式和关联性很有意思。但开始研究生学业之后，我的观点发生了改变，事情变得更加有趣了。

统计学不再是假设检验（结果表明，在许多情况下它并无太大作用）以及寻找模式了。哦，不，我收回这句话。统计学仍然与这些有关，但我对它产生了不一样的感受。统计学其实是在用数据讲故事。我们手头的大堆数据反映了真实的世界，然后我们对它们进行分析，得到的不只是数据的关联性，我们还能了解到身边正在发生什么。这些故事反过来可以帮助我们解决真实世界中存在的问题，例如降低犯罪率、提高卫生意识、改善高速上的交通状况，或者只是增长我们的见识。

很多人都未能找到数据与真实生活之间的联系。我想这也是为什么当我告诉人们我读研是为了学统计学时，大多数人都说那是他们“上学时最痛恨的一门课”。我相信读者们不会犯同样的错误，否则你就不会选择读这本书了，不是吗？

运用数据需要一些技能，如何才能掌握呢？你可以像我一样去学校选择正规的课程训练，但你也可以通过大量的实践经验，自学成才。其实大多数研究生课程和自学也没有多大区别。

在可视化和信息图（infographics）方面也是如此。并不是只有专业图形设计师才能创建优秀的图表，同样，你也不需要拿到统计学的博士学位。你所需要的只是保持对学习的渴望，而且和生活中的所有事情一样，你需要不断练习才能变得更在行。

我制作的第一张数据图大概是在小学四年级，那是为了应付一次课外科学研究。我和搭档一直很想知道蜗牛在什么样的平面上会爬得更快，于是把它们放在各种粗糙或光滑的物体表面上，并计时观察它们爬过一段特定距离各需要多久。最后我拿到了蜗牛在不同表面上爬行的时间数

据，并据此制作了一张柱形图。至于当时是否知道应该将它们按长短进行排序，我已经记不太清了，但是和Excel软件的辛苦纠缠倒是一直刻骨铭心。不过第二年当我们研究赤拟谷盗[①]最喜欢吃哪种谷制品时，作图就是小菜一碟了。当你理顺某款软件的基本功能和操作方式之后，剩下的几乎都轻而易举。这个例子完美地说明了什么叫做从经验中学习。噢，顺便提一句，如果你还在琢磨前面的问题，答案是蜗牛在玻璃上爬得最快，而赤拟谷盗最喜欢吃葡萄果仁麦片（Grape Nut）。

从本质上来说，学习任何软件或编程语言的过程几乎都是一样的。如果你一行代码都没写过，那么R（许多统计学家都采用的一种计算环境）必然会让你望而生畏，而一旦你跟着完成了几个范例之后，就会很快找到窍门。这本书能够帮助你做到这些。

之所以这样说，是因为我本人就是这样学习的。我还记得自己第一次深入接触可视化的设计层面时的情形。那还是我读研究生的第二年，好消息从天而降，我得知自己获得了《纽约时报》图形编辑的实习机会。在那一刻之前，图表对我而言只是一种分析工具而已（比如小学课外活动时作的柱形图），就算其中含有一些美学和设计因素，比重也少得可怜。而将数据用于新闻报道，这对我来说更是无从入手。

所以为了作准备，我阅读了手边能找到的所有设计书籍，以及一本Adobe Illustrator的使用指南，因为我知道《纽约时报》图形编辑部用的就是这款软件。不过还没等我真正上手就已经开始绘制工作了。当你被迫边学边干的时候，就不得不尽快掌握那些必需的知识，而当你开始处理更多数据、设计更多图表时，你的技能也会随之突飞猛进。

如何阅读本书

本书以实例讲解为主，目的是让大家熟悉制图所需的每一个步骤，掌握每一项技能。你可以从头开始完整地读一遍，不过如果你已经有想法在酝酿了，也可以只挑选最感兴趣的几章来读。所有的章节都经过了精心的组织，案例是相互独立的。如果读者对数据领域还比较陌生，那么阅读最开始的几章应该会很有帮助。它们介绍了处理数据的方法、需要关注的重点以及各种可用的工具，便于读者了解如何获得数据，如何规范格式并为可视化作准备。之后的几章会根据不同的数据类型和侧重面分别介绍各种可视化技巧。请记住，永远都要让数据说话。

不管你选择何种阅读方式，我都强烈建议你在阅读时打开电脑，和我一起逐步完成每一个范例，并且浏览在注释和参考中提到的各种资源。你也可以在www.wiley.com/visualizethis或http://book.flowingdata.com[②]上下载到所有的代码、数据文件和可交互演示。

为了表述得更清楚一些，图0-6给出了一张流程图，便于读者找到需要的章节。祝大家阅读开心！

① 赤拟谷盗（red flour beetle）是一种鞘翅目昆虫，对农作物危害极大。

② 亦可从图灵社区（www. ituring.com.cn）本书相关页面下载。——编者注

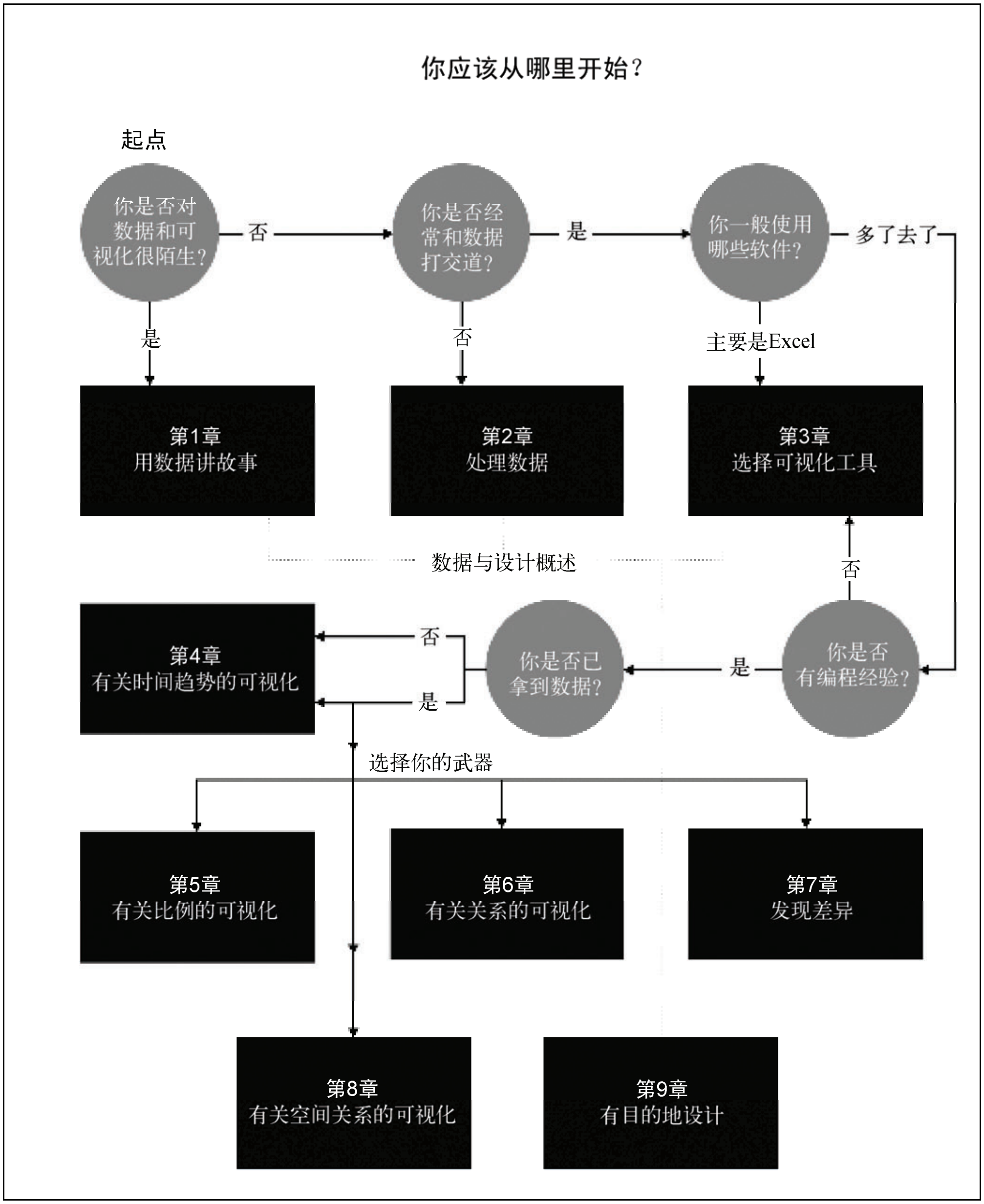

你应该从哪里开始？
起点
你是否对数据和可视化很陌生？
否
是
你是否经常和数据打交道？
是
否
你一般使用哪些软件？
多了去了
主要是Excel
第1章 用数据讲故事
第2章 处理数据
第3章 选择可视化工具
数据与设计概述
否
第4章 有关时间趋势的可视化
否
是
你是否已拿到数据？
是
你是否有编程经验？
选择你的武器
第5章 有关比例的可视化
第6章 有关关系的可视化
第7章 发现差异
第8章 有关空间关系的可视化
第9章 有目的地设计

图0-6　从何处开始阅读

目　录

1 用数据讲故事

让我们先回想一下过去曾看到过的那些数据可视化作品——听演讲时看到的幻灯片、博客文章的配图以及此刻在你脑海中浮现的经典案例。它们的共同点是什么？它们都在讲故事，有趣的故事。这些故事也许是为了让你相信某件事情，或者呼吁你做出某种举动，也可能是通过新的信息激发新思路，或者是打破早已形成的先入之见。不管它们的意图是什么，各种尺寸和形式的优秀数据可视化作品都在帮助我们理解数据背后的意义。

1.1　不只是数字

让我们面对现实——如果在一开始我们不知道自己想了解什么，或者不知道有什么可以去了解，那么数据就是枯燥的。它不过是数字和文字的堆砌，除了冰冷的数值之外没有任何意义。而统计与可视化的好处就在于能帮助我们观察到更深层次的东西。请记住，数据是现实生活的一种映射，其中隐藏着许多故事，在那一堆堆的数字之间存在着实际的意义、真相和美学。而且和现实生活一样，有些故事非常简单直接，有些则颇为迂回费解。有些故事只会出现在教科书里，而其他一些则体裁新奇。讲故事的方式完全取决于你自己，不管你的身份是统计学家、程序员、设计师还是数据研究者。

以上就是我作为一名统计学研究生所学到的第一课。我必须承认在接触这门课之前，我一直认为统计学就是一门单纯的分析学科，而数据也只不过是一种机械式作业的产物而已。的确，在很长一段时间内我都这么想——毕竟，作为一名电气工程专业的大学毕业生，用这种眼光来看待数据是情有可原的。

不要误解我的意思，有这种看法未必是一件坏事。但我在数年后认识到，数据虽然有很强的客观性，但其中往往也存在着人的因素。

比如说，让我们再来看看失业率问题。说出一个全国平均数字很容易，但正如我们所见，不同地区对每个人的意义是不一样的，甚至连街区的远近都会造成差异。如果你认识的某个人这两年一直没有工作，其实是很难把他视为一个冷冰冰的统计数字的，不是吗？这些数字代表的是活生生的人，所以我们处理数据时必然会带有感情因素。当然，我们也不可能讲述每一个人的故事，但在“失业率攀升5个百分点”和“数十万人失业下岗”之间还是有着微妙但重要的区别。前者读起来只是一个没有多少上下文背景的数字，而后者却更能让人产生共鸣。

1.1.1　新闻报道

在《纽约时报》的图形设计实习机会让我感触颇深。虽然那只是我研二暑假期间的短短三个月而已，但它影响了我此后处理数据的方式。我学会的不仅是为新闻绘制图表，我还学会了如何像报道新闻那样报道数据，而这需要更多的精力去设计、组织、核查、追踪和研究。

有一次我的任务是核实某个数据集中的三个数字，因为《纽约时报》的图形编辑部在绘制图表时，必须确保报道的所有数据都是准确无误的。我们只有在确信所有数据都可靠之后，才会去考虑表现层面的问题。正是这种对细节的关注让该报的图表如此优秀。

《纽约时报》的任何一个图表都是如此。它将数据表现得非常清晰、简洁，甚至可以说是优美。这代表着什么呢？只有在阅读这样的图表时，你才能理解数据的意义。重要的点或区域都带有注释，使用的符号和颜色都进行了详细的说明，而且它能让读者轻松地了解数据中的信息。这已经不仅是图形了，这简直是图形的艺术。

图1-1就类似于你在《纽约时报》中所看到的那些图表。它显示了各年龄段的人在未来一年内死亡的概率。

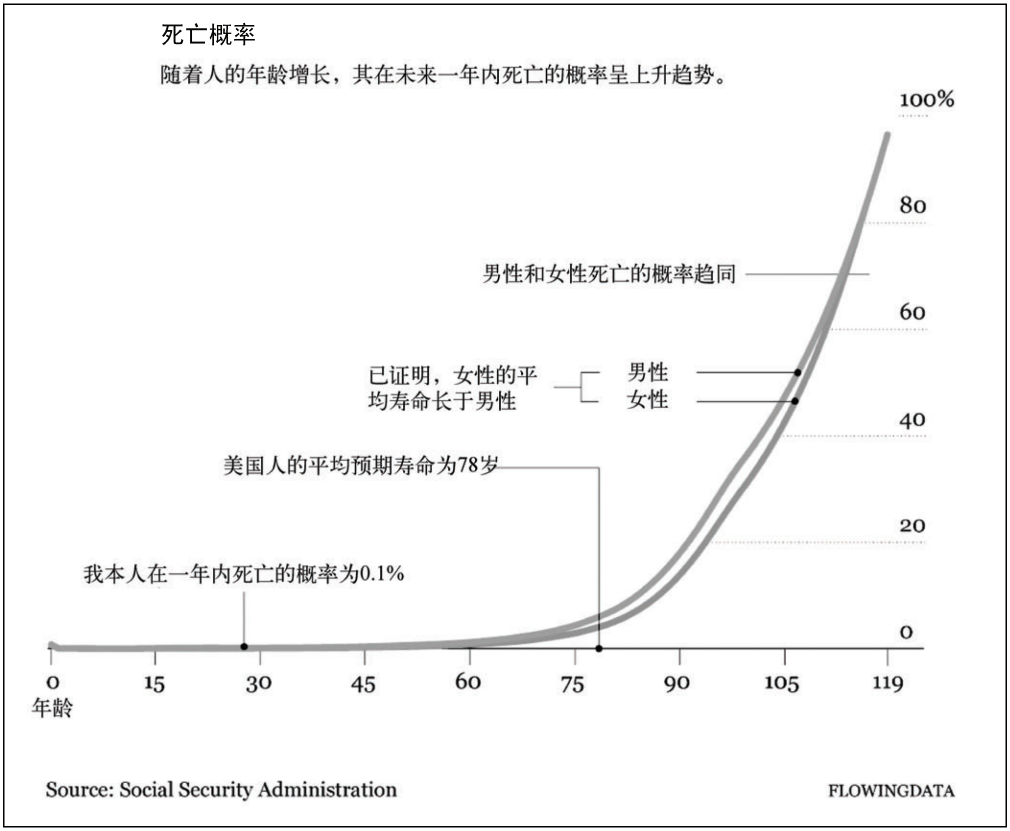


图1-1　不同年龄段的死亡概率

▶访问http://datafl.ws/nytimes，欣赏《纽约时报》的一些优秀图表。

这张图表从根本上来说只是一个折线图（line chart）而已，但是其中的设计元素让它显得生动：指针和标记提供了上下文环境，便于读者理解这些数据的有趣之处，而曲线的颜色和宽度则将他们的视线引导到重要的地方。

图表和图形并不只是将统计结果可视化，它们还对可视化展现的内容进行解释。

说明　观赏Geoff McGhee的纪录片*Journalism in the Age of Data*（数据时代的新闻业），了解新闻记者是如何运用数据来报道当今事件的。片中采访了许多业界高人。

1.1.2　艺术

《纽约时报》是客观的。它为人们呈现数据、展现事实，而且干得很棒。但可视化的另一面

则不太注重分析，而更多偏重于挖掘人类的情感，比如Jonathan Harris和Sep Kamvar的作品*We Feel Fine*（见图1-2）。



图1-2　Jonathan Harris和Sep Kamvar的作品*We Feel Fine*（另见彩插图1-2）

这件交互作品可以从各个公开的个人博客中抓取词句，然后将它们以悬浮气泡的形式展现出来。每一个气泡都代表着某种情绪，而且有相应的颜色标记。从整体来看，气泡就像无数个体在空间中无止境地漂浮，但观察一段时间之后你就会发现它们开始聚集。如果在顶部菜单中选择各种分类，还能看到这些貌似随机的片断之间的联系。点击单独的气泡可以看到它自身的来龙去脉。整个作品既富有诗意又能给人以启迪。

▶访问http://wefeelfine.org，欣赏Jonathan Harris和Sep Kamvar的在线作品，并探索人们此刻的情绪。

还有很多类似的例子，比如Golan Levin的*The Dumpster*（垃圾桶），它同样也是抓取博客，但只涉及与他人断绝关系的内容。还有Kim Asendorf的*Sumedicina*，讲述了一个人脱离腐败组织的虚构故事，里面没有只言片语，只有图形图表。这样的例子还包括Andreas Nicolas Fischer展现美国经济滑坡的实体雕塑。

▶访问http://datafl.ws/art，观看FlowingData 上更多有关数据和艺术跨界的例子。

重点是，数据和可视化并不一定只能和冰冷的、不争的事实相关。有时我们寻求的并不是分析和洞察，而是从富有情感的观点来讲述故事，鼓励读者对数据作出回应。并不是所有的电影都必须是纪录片，同样，也并不是所有的可视化都必须是传统的图形或图表。

1.1.3　娱乐

在新闻报道和艺术创作之间，可视化在娱乐领域也找到了一席之地。如果我们以更为抽象的角度来看待数据，在电子表格和带分隔符的文本之外，将照片和状态更新也包括在内，那么就不难发现这一点。

Facebook利用人们的状态更新来判定一年中最快乐的日子是哪一天，而在线约会网站OkCupid则根据用户的在线信息来估算人们会怎样夸大自己在数字世界中的形象，如图1-3所示。这些分析并不会为网站带来业绩的提升、收入的增加，也不可能帮助排除系统故障，但它们却因为娱乐方面的价值而像野火一样迅速在互联网上四处传播。数据能在一定程度上折射出我们自身和社会的投影。

Facebook发现一年中最快乐的日子是感恩节，而OkCupid发现人们一般都会将自己的身高夸大2英寸（约5厘米）左右。

▶ 访问OkCupid的博客OkTrends（http://blog.okcupid.com）以查看有关网上约会的更多发现，例如白色人种最喜欢的是什么，以及如何在细节处保持形象。

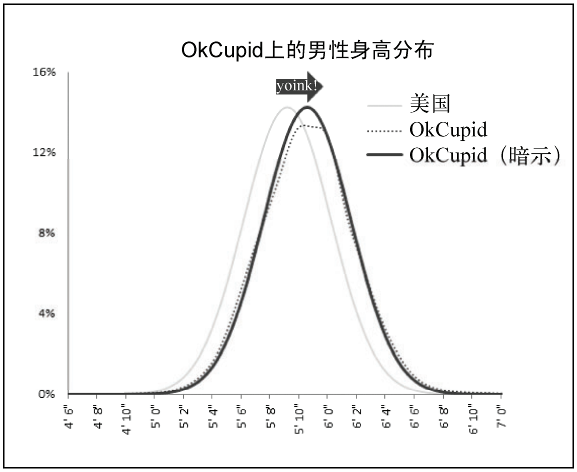


图1-3　OkCupid上的男性身高分布

1.1.4　引人注目

当然，并不是所有的故事都以增长见闻或者娱乐大众为目的。有时候它们是为了提出倡议或者呼吁人们作出行动。在纪录片《难以忽视的真相》中就有令人难忘的一幕：主讲人Al Gore要依靠剪叉式升降机的帮助才能靠近二氧化碳的上升曲线。

不过在我看来，卡罗琳学院的国际卫生学教授、Gapminder基金会理事Hans Rosling在这方面才是真正的无人能及。Rosling曾经运用一款叫做Trendalyzer的工具（见图1-4），以动画的方式来展现世界各国摆脱贫困的历程。在他的那次演讲中，所有观众从一开始就被深深吸引到数据的世界里，而在结束时都情不自禁地起立鼓掌喝彩。真的非常精彩，强烈建议不要错过。

▶欲知Hans Rosling如何用数据和精彩的演示赢得现场观众的喝彩，访问http://datafl.ws/ hans。[①]

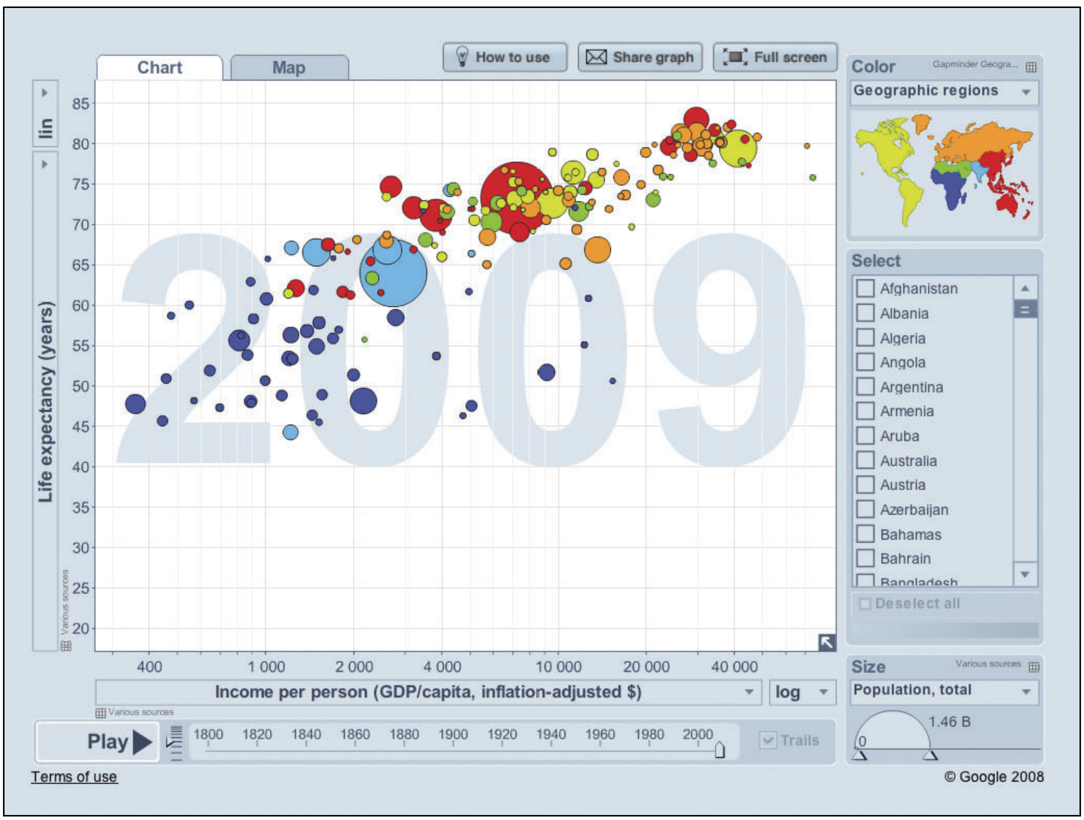


图1-4　Gapminder基金会开发的Trendalyzer软件（另见彩插图1-4）

① 可选择中文演讲稿。

他运用的可视化技巧非常简单，不过是一种可运动的图表而已。其中的气泡代表各个国家，根据该国的贫富程度在时间轴上移动。那他的演讲为何会如此大受欢迎呢？因为Rosling在演讲时充满着信念和激情。他是在给大家讲故事。我们都遇到过配有图形图表演示的演讲者，但其中绝大多数都让观众昏昏欲睡。而Rosling却把握住了数据背后的意义，并将其作为自己的优势，最后的吞剑表演则更是水到渠成的一笔。在看了Rosling的演讲之后，我忍不住想亲手把玩那些数据，重温一遍他所讲的故事。

之后我又看过Gapminder基金会的另一次演讲，相同的主题、相同的可视化方式，但却是另一位演讲人。这次就不那么令人兴奋了——说老实话，称其为催眠曲也不为过。在数据中观众感觉不到任何情绪，也不能让人从中感到任何信念或激情。由此可见，数据本身或许并不一定趣味盎然，令人产生深刻印象的是你设计数据和演示数据的方法。

综上所述，我们应当以讲故事的角度来思考如何可视化。你打算讲一个什么样的故事呢？它的体裁类似于报告还是小说？你是否想说服人们应当采取某种行动？

和小说中的人物成长一样，每一个数据背后都有它自己的故事，如同书中的每一个角色都有着各自的过去、现在和未来。那些数据彼此之间充满着互动和联系，如何表现出这些关系则取决于你自己。这就像在动笔写小说之前，我们必须先学会遣词造句一样。

1.2　我们要寻求什么

要讲故事，没问题。那么，到底应该怎样用数据来讲故事呢？自然，具体细节会因为数据的不同而有所差别，不过总体来说，不管图形表现的是什么，我们都需要留心观察以下两件事情：模式和相互关系。

1.2.1　模式

事物会随着时间的流逝而变化。我们都会老去，青丝变白发，视力不复从前（见图1-5）。产品的价格会上下浮动，品牌的商标逐年更换，各种行业新老交替。变化有时突如其来、毫无征兆；有时则十分缓慢、难以察觉。

无论观察的对象是什么，变化本身可以和变化的过程一样妙趣横生。这里探究的正是随时间推移而变化的各种模式。比如在观察股票价格时，它们自然会有增有减。但每天的变化量是多少？每周或每个月的变化量又是多少？是否在某个时期内股票的波动会异于往常？如果是，其原因何在？是否有某些特殊的事件引发了这些变化？

如你所见，由某一个问题出发会带来更多的问题。并不是只有时间序列数据会这样，所有类型的数据都是如此。努力探索数据背后的含义，你就会发现更多有趣的答案。

我们也可以用不同的方式来拆分时间序列数据。有时需要按天或小时来显示数值，有时则更适合以年或月为单位。前者的时序图会显示出更多杂点，而后者则偏向于呈现总量。

使用过网站分析工具的人可能会比较认同这一点。以日为单位来监测网站流量时，图表就会起伏不平，存在很多波动，如图1-6所示。

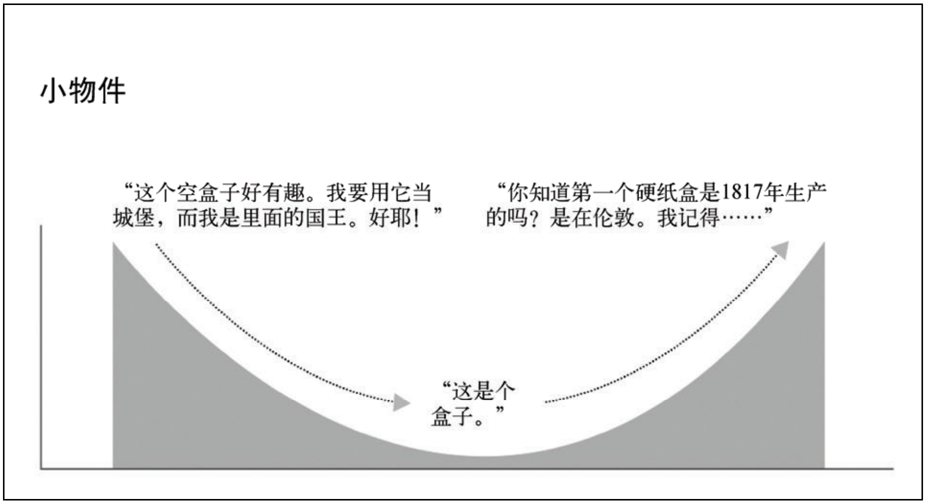


图1-5　喜剧化地看待衰老

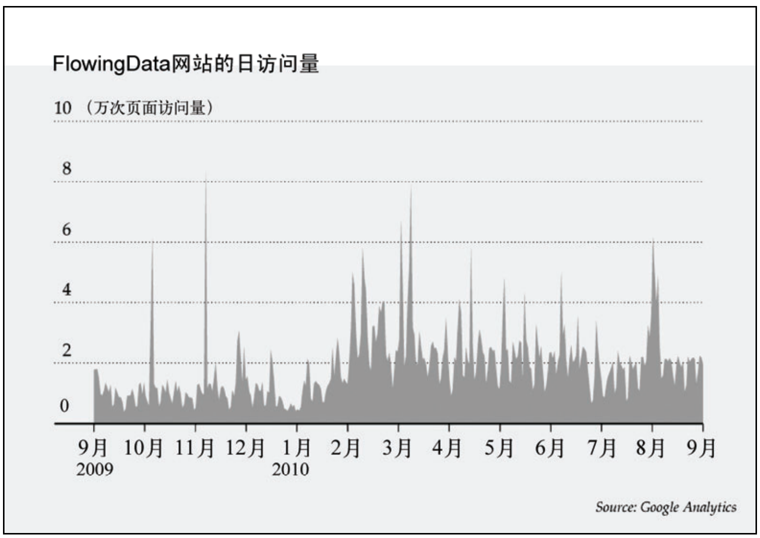


图1-6　FlowingData网站的每日独立访客量

而当我们以月为单位来观测时，图表上涵盖同样时间跨度的数据节点就会减少，显得更加平滑，如图1-7所示。

我并不是说某个图表比另一个要好。事实上它们是相互补充的。如何拆分数据则取决于你需要（或不需要）多少细节。

当然，我们寻求的不仅是随时间而变化的模式。还有很多有关总量的模式能帮助我们对团体、人或事进行比较。你每周会吃什么食物、喝什么饮料？美国总统在每年发表国情咨文时通

常都会谈论哪些内容？有哪些州倾向于投共和党的票？对于后面这个问题，按地区性来寻找模式显然会更有帮助。尽管问题和数据类型各有不同，但我们的处理方式是相似的，这在后面几章中就会看到。

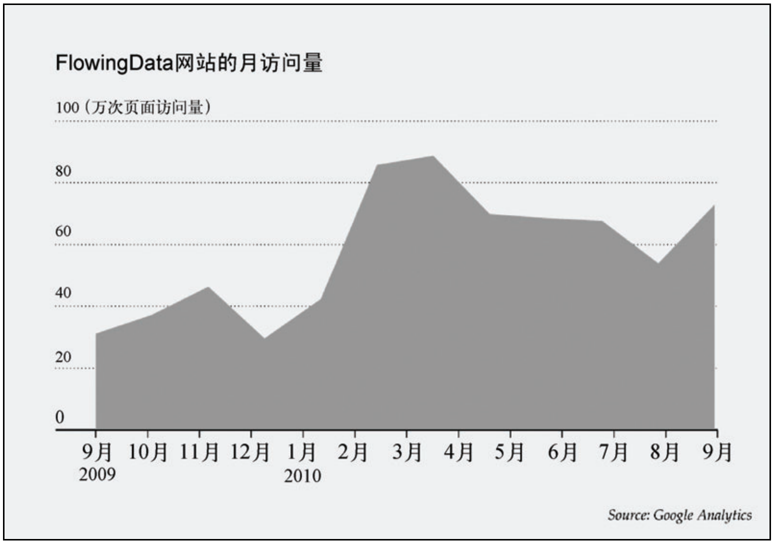


图1-7　FlowingData网站的每月独立访客量

1.2.2　相互关系

你是否见过那种包含了一大堆图表、看上去像是随意堆砌的数据图？这种图表似乎丢失了某种特别的东西，就好像是设计师敷衍了事、赶在最后期限到来之前匆匆完成的作品。一般来说，这种特别的东西就是图表相互之间的关系。

在统计学中，它通常代表的是关联性和因果关系。多个变量之间应该存在着某种联系。第6章将详述这些概念以及可视化的方法。

不过在更为抽象的层面，抛开各种等式或假设检验不谈，我们完全可以在视觉上对数据图进行设计，用于比较和对照各种数值和分布。这里有一个简单的例子，摘自我的作品*World Progress Report*（世界发展报告）中有关科技的部分，如图1-8所示。

> ▶ *World Progress Report*是一份有关全球发展对比的图形化报告，数据来源于UNdata。请访问http://datafl.ws/12i浏览完整版本。

我使用了直方图来表现每100个居民中互联网用户、互联网订购者和宽带用户的数量。请注意互联网用户的跨越幅度（从没有用户到每100居民中有95位用户）比其他两个数据集要宽得多。

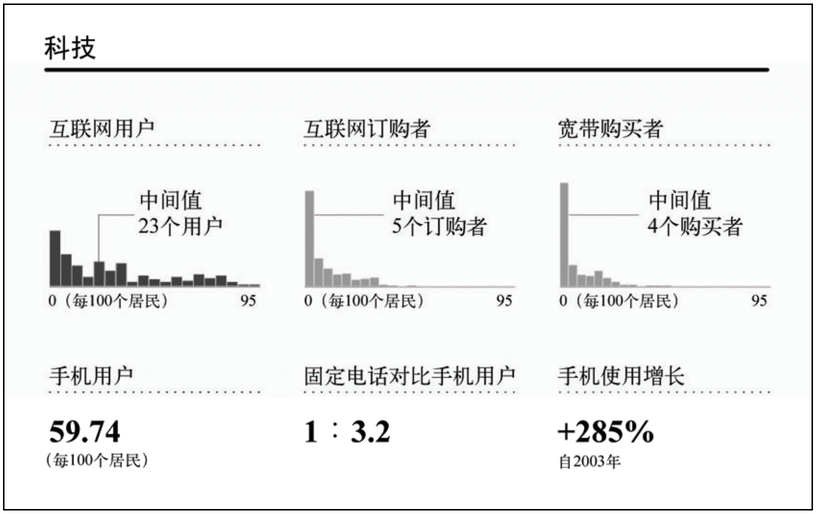


图1-8　全球范围的科技采用

最方便的办法自然是让软件根据数据的多少来自动设定每一个直方图的幅度。但是，尽管没有任何国家的互联网订购者或宽带用户达到95%，我仍然赋予每一个直方图以相同的幅度。这能让读者一目了然地比较各群体间的分布情况。

当你需要处理很多不同的数据集时，请尝试将它们进行分组，而不是当做互不干涉的单独元素来看待。这样能够产生更加有趣的结果。

1.2.3　有问题的数据

在数据中寻找故事时，我们应该对自己所看到的保持质疑态度。记住，决不能只因为它是数字就相信它就是正确的。

我必须承认，数据检验无疑是我在数据图制作过程中最不喜欢的一步。我的意思是说，当某个人、组织或服务机构为你提供一大堆数据时，本应该由他们来确保所有数据都是真实可信的。但这同样也是一位优秀数据图形设计师的分内之事。毕竟，可靠的建筑师不会用劣质水泥来为房屋奠基，所以我们也不要用劣质的数据来构建自己的数据图。

也许数据检验与核实不是数据图设计中最重要的部分，但它绝对是其中不可或缺的部分之一。

基本上，我们要注意的是那些不太对头的东西。也许是因为在数据输入时出现了失误，某人多添或者遗漏了一个零；也许是在数据收集时网络出现了问题，有些字节被随机移动到了其他地方。无论原因是什么，如果任何东西看上去有些异常，我们就需要到源头去进行验证。

提供数据的人通常都会对数据所表现的普遍状况有所认识。如果你自己就是收集数据的人，自问一下这是否说得过去：所有地区在某方面的指标都只有10%～20%，而某个地区却达到了90%。那里是不是出了什么问题？

大部分异常都只是笔误而已，但有些异常却真的存在，而它们就是有意思的地方，可以作为故事的重点。如果你遇到了异常，一定要确定它到底属于前者还是后者。

1.3　设计

当所有的数据都安排妥当，就可以着手进行可视化了。无论你做的是什么，报告、网上的信息图抑或是数据艺术品，都应当遵循一些基本的规则。所有这些规则都是弹性的，更像是一个框架而不是生搬硬套的教条，但如果你刚刚接触数据图形，从这些开始应该不会错。

1.3.1　解释编码

数据图形的设计流程都是相似的。你先拿到数据，然后以圆形、柱形和颜色等形式对数据进行“编码”，最后呈现给读者。因此读者必须先对你的编码进行“解码”：这些圆形、柱形和颜色分别代表什么？

William Cleveland和Robert McGill曾撰文对编码进行过详细的探讨。某些编码方式要比其他编码方式更行之有效。但如果一开始读者不清楚你的编码所代表的对象，那么选择任何编码方式都没有效果。如果他们不能解码，那么你对数据图的辛苦设计就是无用功。

说明　阅读Cleveland和McGill所著论文“Graphical Perception and Graphical Methods for Analyzing Data”（数据分析的图形化感知和图形化方法），以深入了解人们如何编码形状及色彩。

有时候我们会在数据艺术品或信息图中看到这种图形缺失上下文的情况。可能在数据艺术品中更为常见。过多的标记或说明文字确实可能会破坏作品的气氛，但至少可以在旁边给出一小段描述性的文字，提供必要的信息。这能帮助其他人理解并欣赏你的设计成果。

如果在真正的数据图中出现这种情况，读者就会感到沮丧失望，而这是我们最不愿意看到的。有时候这是因为设计师一直与数据打交道，他们知道每一样东西都代表什么意思，所以在设计时忘了面面俱到。但是读者在看到图表之前对它是一无所知的，他们并不熟悉设计师在分析过程中所了解到的上下文。

那么如何确保读者能够解码我们的编码呢？可以通过标记、说明文字和图解来解释各个图表所代表的内容。选择何种解释方式取决于具体情况。比如说，让我们看看图1-9中的世界地图，它显示了Firefox浏览器在各个国家的使用情况。



图1-9　世界各国Firefox的使用情况（另见彩插图1-9）

我们可以看到不同国家蓝色有深浅之别，但这代表什么意思呢？深蓝色是代表使用量大还是小？如果深蓝色代表使用量大，那么大的衡量标准又是多少？如果只有这些元素，这张地图对我们来说几乎没有多少作用。但如果设计师提供了图1-10中的说明，事情就会清楚很多。除了对色彩进行了说明外，它还是一个直方图，根据用户数量的多少表现了使用情况的分布。

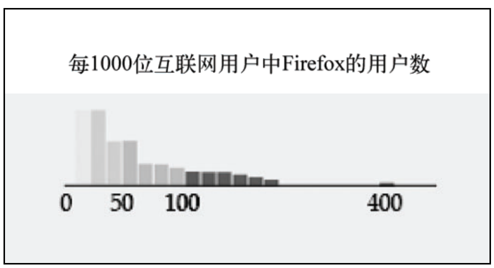


图1-10　Firefox使用区域数据图的说明（另见彩插图1-10）

如果有足够的空间，而且分类不多的话，我们也可以直接在形状或对象旁边添加标记，如图1-11所示。这个图表显示了明星们在最终捧得奥斯卡最佳男主角奖之前在往届曾获得的提名次数。

网上流传有一种理论，在所有入围者中，往年获得提名次数最多的人通常就会赢得当年的金像奖。如图中所标注的，暗橙色（图中为深色）表示该获奖者确实是往届提名最多的人，而亮橙色（图中为浅色）则表示这位大奖得主曾获得的提名次数并不是所有入围者中最多的。[①]

如你所见，要想对图表进行解释，我们可以有很多选择。运用它们的方法都很容易，但正是

① 出于印刷成本的考虑，在不影响读者理解的前提下，本书中一些色彩简单的图未给出彩插，而通过灰度的深浅来区分。——编者注

这些细节之处造就了数据图的不同阅读方式。



图1-11 直接对对象进行标注

1.3.2 标注坐标轴

对编码需要进行解释，对坐标轴也应当进行标注。没有标注或解释的坐标轴就只是个摆设而已。标注坐标轴可以让读者知道它所描绘的尺度和内容，否则就会猜测它代表的到底是增量、对数、指数，还是每100个抽水马桶如何如何。就我个人来说，每当不清楚坐标轴的标记是什么时我就会假设它代表的是最后一种意思。

数年前我曾在FlowingData上举办过一次竞赛。我发表了如图1-12所示的一张图片，然后邀请网友来标注坐标轴，看谁标注的结果最有乐趣。

针对同一幅图形我收到了大约60份不同的标注。图1-13中显示了其中的一部分。

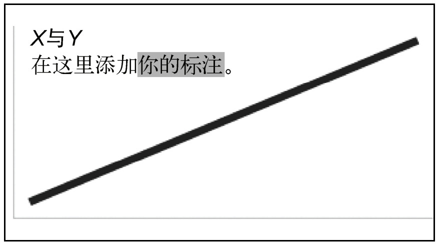


图1-12　在这里添加你的标注

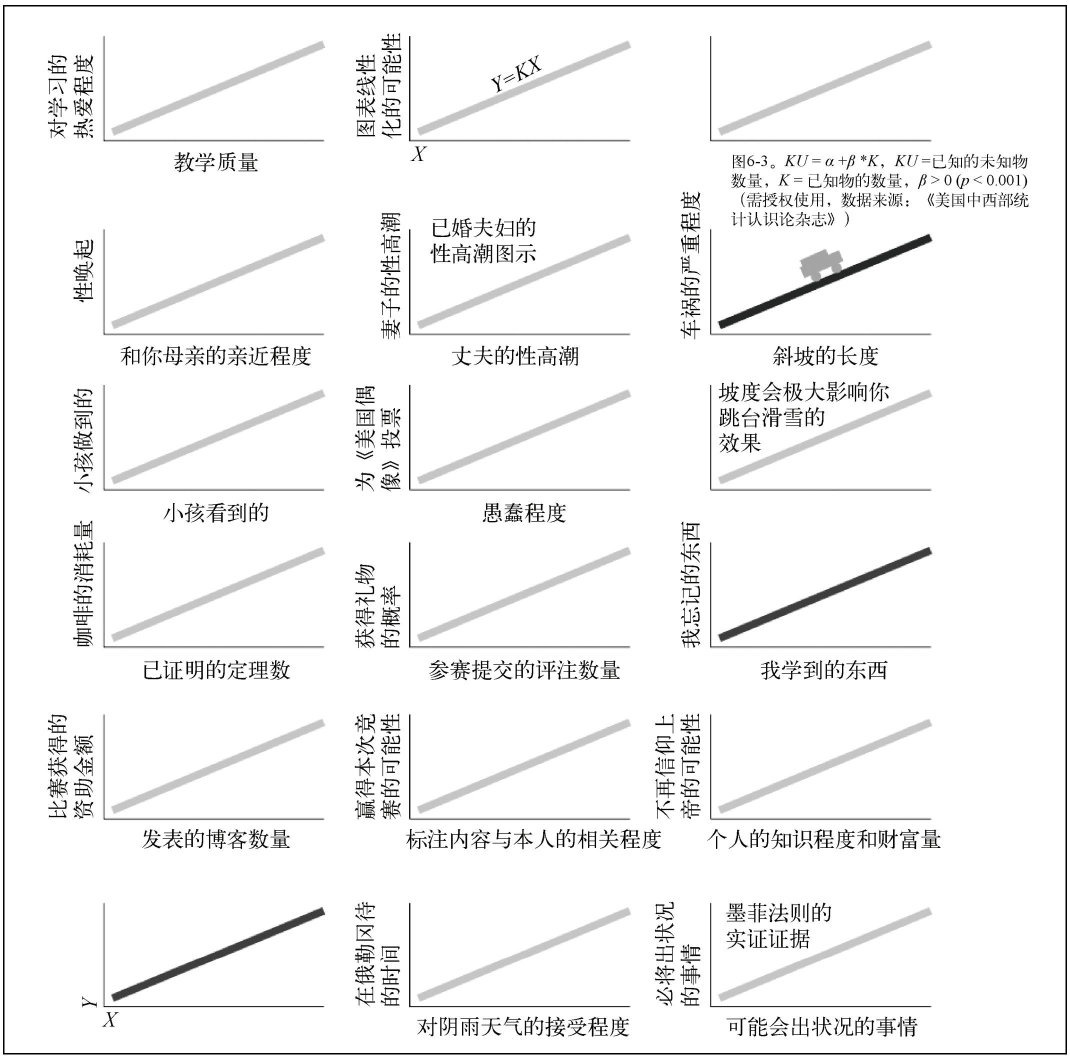


图1-13　FlowingData标注竞赛的部分投稿作品

如你所见，尽管每个人看到的都是同一幅图表，但只要改动坐标轴的标记就会得到一个完全不同的故事。当然，这只是为了好玩，但试想如果你想让别人认真对待你的图表，情况又会怎样？没有了标记，你的图表就毫无意义。

1.3.3　确保几何上的正确性

在设计数据图时必定会用到几何形状。柱形图中有矩形，它们的长度代表了数值的多少。在点状图中表现数值的则是位置，这和标准的时序图类似。饼图通过角度来表现数值，其总和通常都等于100%（参见图1-14）。这些都很简单，但是也很容易出错，所以要多加小心。如果我们不加注意就有可能弄糟，而读者尤其是网友们对此是不会留情面的。

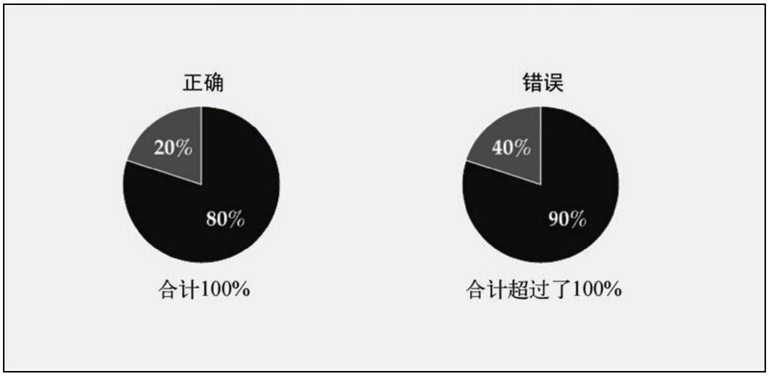


图1-14　绘制饼图的正确及错误方法

另一个常见的错误是，设计师运用二维的形状来表现数值，但却把它们当做是一维来比较。柱形图中的矩形虽然是二维的，但我们只会用到它的长度，宽度其实并无实际意义。但如果是气泡图，就会用面积来表现数值，而新手往往只会考虑半径或直径，导致比例完全错误。

图1-15中的一对圆形根据面积大小来计算对比关系。这是正确的方法。

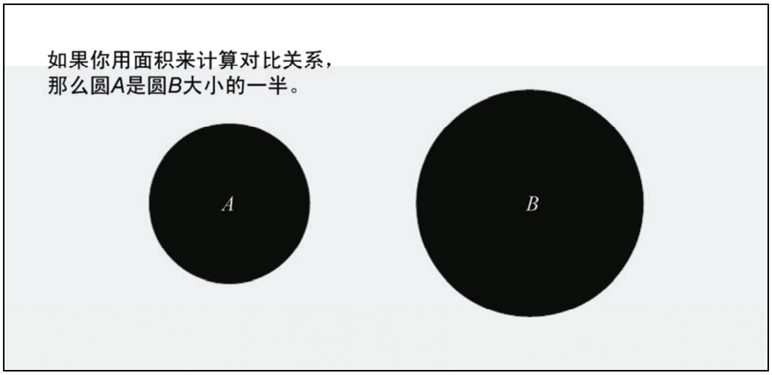


图1-15　比较气泡的正确方法

图1-16中的一对圆形则是根据直径来计算的。第二个圆的直径是第一个的两倍，但面积却是它的4倍。

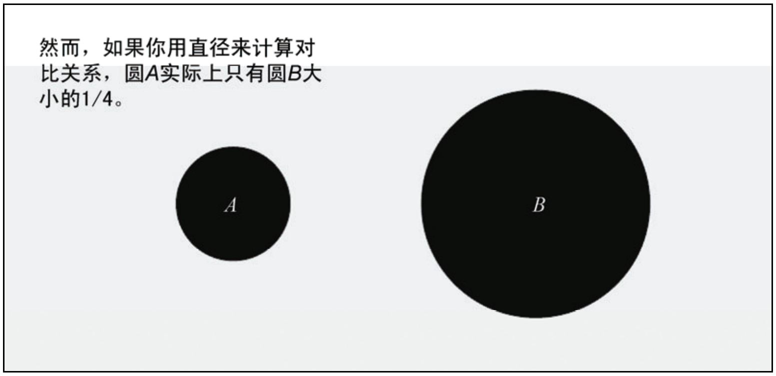


图1-16　比较气泡的错误方法

在处理矩形时也是一样，例如板块层级图（treemap）。我们用矩形的面积来代表数值，而非仅取用它的长或宽。

1.3.4　提供数据来源

本来这一点毋庸赘言，但就是有很多人都漏了它。你的数据是从哪儿来的？我们在报刊杂志上看到的数据图都附有数据来源，通常都以小字体印在图表的底部。你也应该如此做，否则读者就没办法知道你的图表有多准确。

没有数据来源，读者也可能会怀疑图中的数据是不是胡编乱造的。当然我们不会这么做，但并不是所有人都会相信这一点。除了为图表带来信誉之外，数据来源还能便于读者进行核验或者分析。

提供数据来源还能为数据带来更多的上下文背景。在某个小镇上搞的民意测验与全美人口普查相比，得到的诠释自然会有天壤之别。

1.3.5　考虑你的受众

最后，多考虑数据图本身的设计意图和受众群体。比如说，演讲幻灯片中的饼图就应该设计得尽量简单明了。如果在其中加入大量细节，恐怕就只有坐在前排的听众才能够看清。但如果是供人学习研究用的印刷品，添加细节就没太大问题。

如果数据图是放在商业报告里，那么就不要把它设计成惊世骇俗的艺术作品。简单、直达主题才是你追求的目标。如果只是用于分析，那么数据图本身就够了，无需在美感和注释上花费大量时间。如果你的图形是为了发布给大众，那么设计就不要过于复杂，同时为任何可能产生疑义的概念提供解释。

1.4　小结

简而言之，以某个问题作为出发点，以批判的眼光检查你的数据，并且把握图形的设计意图以及受众群体。这有助于你设计出清楚的数据图，不会让读者失望——不管是哪方面的图形都是如此。

在后续几章中我们会谈到具体的做法。你将学会如何处理数据并且进行可视化，如何从头到尾设计出完整的数据图。之后你就能学以致用——找出你自己要讲的故事，然后进行相应的设计。

2 处理数据

在我们开始视觉化之前，首先需要有数据。正是数据让可视化有趣起来。如果数据本身没有意思，就只能是一张好看但无用的图片而已，很快就会被人丢在脑后。那么在哪里才能找到好的数据？又怎样去获得它呢？

在拿到了数据之后，我们还需要对它们进行格式化，以便输入到软件中去。你得到的可能是以逗号分隔的文本文件，或者Excel电子表格，而你需要将它们转换成像XML这样的语言。或者反之亦然：也许原始的零散数据来自于某个网页应用，但你需要的是一份完整的电子表格。

首先学会如何获得数据和处理数据，然后再提高我们的可视化技能。

2.1 收集数据

数据是任何可视化的精髓与核心。幸运的是，有很多途径可以找到它们，例如求助于领域内的专家，使用各种在线应用，或者靠自己的力量来收集。

2.1.1 由他人提供

这种途径非常普遍。如果你是自由设计师，或者是在大公司图形部门工作的设计师，那么就更是如此。这通常是一件好事，因为有人为你分担了数据收集这部分工作，但仍旧不能大意。数据在变成格式优美的电子表格呈现给我们之前，途中可能会出现很多错误。

在我们用电子表格来分享数据时，最常见的错误就是笔误。有没有漏掉零？这里是否应该是6而不是5？一般来说，除了带分隔符的文本文件之外，数据都是直接从来源读取，然后导入到Excel或其他电子表格程序里面，因此一些小的笔误很容易就会蒙混过关，来到我们手中。

上下文环境也需要检查。我们不必成为研究数据主题的专家，但至少应该知道数据的原始出处、收集的过程以及背后的主旨。这能帮助我们创建出更加优秀的数据图，让故事更加完整。假如你关心的是问卷调查，那么调查是何时举办的？是谁发起的？有哪些人参与？很明显，20世纪70年代的问卷调查结果和今天的必将截然不同。

2.1.2 寻找数据源

如果没有人提供数据，我们只能自己花力气寻找。从坏的一面来看，肩上的担子更重了，但从好的一面来看，找到相关度高的数据会越来越容易，而且机器可读性也会更高（也就是说，更方便输入到软件中去）。我们可以从以下方面着手。

1. 搜索引擎

今天我们怎样在网上找东西？我们用Google。这几乎是顺理成章的事情，但仍然有很多人发邮件问我应该怎样才能得到某某数据、是否有什么便捷方法。我个人一般去的就是Google，偶尔也会去Wolfram|Alpha（这是一款带有计算能力的搜索引擎）。

> ▶访问http://wolframalpha.com试用Wolfram|Alpha。如果你需要某方面的基础统计结果，这款搜索引擎会非常有帮助。

2. 直接数据源

如果直接查询“数据”不能得到任何有用的结果，可以尝试求助于该领域的学者。有时候他们会在其个人网站上发布数据。如果没有，也可以翻阅他们的论文或学术报告寻求线索。你甚至还可以直接给他们发邮件，但要先确定他们确实作过相关的研究，否则只会是浪费大家的时间。

你也可以在《纽约时报》等新闻机构发布的图表中寻找数据源。这些来源通常都会以小字体附在图表的某处。如果这些地方没有，相关文章中也应该会提到。如果你在报纸或网上看到某个图表正好使用了你感兴趣的数据，这一招会很有用。搜索这个来源所属的网站，也许就能得到完整的数据。

自称是某某报纸的记者，直接向文章作者发邮件也是一条路。不过还是先试试能不能找到来源网站吧。

3. 大学资源

作为一名研究生，我常常会利用学术资源，也就是大学图书馆。许多图书馆都扩充了它们的科研资源，拥有丰富的数据存档。一些统计学院系还登记了数据文件清单，其中有很多都对公众开放，虽然许多院系的数据库是为课程实验室和在籍学生的练习作业准备的。我建议访问以下资源。

- 数据及故事图书馆（Data and Story Library，DASL，http://lib.stat.cmu.edu/DASL/）——有关数据文件以及讲述基础统计方法用法的在线图书馆，来自卡内基梅隆大学。
- 伯克利数据实验室（Berkeley Data Lab，http://sunsite3.berkeley.edu/wikis/datalab/）——加州大学伯克利分校图书馆系统的一部分。
- 加州大学洛杉矶分校统计数据库（UCLA Statistics Data Sets，www.stat.ucla.edu/data/）——加州大学洛杉矶分校统计学院的数据库，主要用于实验室和课程练习。

4. 综合数据类应用

有关数据提供的综合性网络服务日益增多。有些网络应用提供了大型的数据文件，供人有偿或无偿下载。还有一些应用则由广大开发人员创建，通过应用编程接口（Application Programming Interface，API）获得数据。这能让我们运用某些服务应用（例如Twitter）的数据，并整合进自己的程序中去。以下是其中一些资源。

- Freebase（www.freebase.com）——一个主要致力于提供关于人物、地点和事件的数据的社区。它在数据方面有点类似维基百科，但网站的结构更清晰。可以下载网友上传的数据文件，或者将你自己的数据进行备份。
- Infochimps（http://infochimps.org）——数据市场，提供免费和收费的数据下载。你也可以通过他们的API来获得数据。
- Numbrary（http://numbrary.com）——为网上的数据进行编目，主要为政府数据。
- AggData（http://aggdata.com）——提供付费的数据集，多关注于各种零售业的地区性数据。
- 亚马逊公用数据库（Amazon Public Data Sets，http://aws.amazon.com/publicdatasets）——更新不多，但确实有一些科研方面的大型数据集。
- 维基百科（http://wikipedia.org）——在这个靠社区运转的百科全书中有大量HTML表格格式的小型数据集。

5. 专题性数据

除了综合性的数据提供商之外，还有很多主题较单一的网站，它们提供了大量免费的数据。

以下是按部分主题进行的分类。

- 地理

只有绘制地图的软件，但却没有地理方面的数据？你走运了。有大量的形状特征文件和地区性数据资料任你调用。

- ❑ TIGER（www.census.gov/geo/www/tiger/）——来自美国人口统计局，可能是目前最全、最详细的有关道路、铁路、河流及邮政区域等方面的数据。
- ❑ OpenStreetMap（www.openstreetmap.org/）——最好的数据社区之一。
- ❑ Geocommons（www.geocommons.com/）——既有数据，又有地图绘制软件。
- ❑ Flickr Shapefiles（www.flickr.com/services/api/）——根据Flickr用户上传照片获得的地理数据。

- 体育

人们热爱体育竞技方面的统计，近几十年来的竞技数据都不难找到。你可以在《体育画报》等杂志或者各球队官方网站上找到它们，也可以去专门的数据型网站。

- ❑ Basketball Reference（www.basketball-reference.com/）——提供每一场NBA赛事的详细数据。
- ❑ Baseball DataBank（http://baseball-databank.org/）——可以下载到美职棒联赛完整数据的入门级网站。
- ❑ databaseFootball（www.databasefootball.com/）——可浏览全美橄榄球联盟（NFL）所有球队、球员和赛季的数据。

- 全球

一些大的国际性组织都有关于全球性的数据，主要集中在卫生保健和发展指标等方面。不过需要筛选一下，因为大部分数据都相对稀疏。在各个国家的数据间建立统一的衡量标准也不太容易。

- ❑ 全球卫生事实数据库（Global Health Facts，www.globalhealthfacts.org/）——世界各国医疗卫生方面的数据。
- ❑ UNdata（http://data.un.org/）——来源众多的全球数据聚合。
- ❑ 世界卫生组织（World Health Organization，www.who.int/research/en/）——同样是医疗卫生方面的数据，例如死亡率及平均寿命。
- ❑ 经合组织统计（OECD Statistics，http://stats.oecd.org/）——各国经济指标数据的主要来源。
- ❑ 世界银行（World Bank，http://data.worldbank.org/）——数百种指标数据，而且便于调用。

- 政府与政治

近年来开始强调数据的透明公开，因此许多政府机构都公布了数据，而类似阳光基金会（Sunlight Foundation）这样的组织也鼓励开发和设计人员对其加以利用。自从data.gov网站启动后，很多政府数据被集中到了一处。我们还能找到许多对政治家起到舆论监督作用的非官方机构网站。

- ❑ 美国人口统计局（www.census.gov/）——大量的人口统计资料。
- ❑ Data.gov（http://data.gov/）——为政府机构提供的数据进行编目。相对还比较新，但拥有很多资料来源。

- Data.gov.uk（http://data.gov.uk/）——英国的Data.gov。
- DataSF（http://datasf.org/）——专门提供旧金山市的相关数据。
- NYC DataMine（http://nyc.gov/data/）——和DataSF相似，不过对应的是纽约市。
- Follow the Money（www.followthemoney.org/）——大量工具和数据集，主要用于监督、调查美国政府的开支。
- OpenSecrets（www.opensecrets.org/）——同样提供政府在竞选等方面花销的详细数据。

2.1.3　自动搜集数据

通常我们都能找到需要的数据，但有一个问题会很麻烦，那就是它们都不在同一个地方、同一个文件里，而是散落在多个网站、多个HTML页面中。这时候应该怎么办呢？

最简单直接、但也最耗时的方法就是访问每一个网页，把感兴趣的数据手工输入到电子表格中。如果你需要的只是几个页面，这当然没什么大不了的。

但如果有几千个页面呢？这种情况要花的时间可就长了，就算只有一百个页面也会让人难以忍受。如果这个过程能自动完成，事情就会轻松得多，而这正是“自动搜集”的含义所在。通过一点代码，程序就能自己访问大堆页面，从中抓取需要的内容并存储到我们的数据库或文本文件中。

说明　*在搜集数据时，使用代码自然是最灵活的方式，但也不妨使用类似Needlebase或者Able2Extract PDF转换器这样的工具。它们的用法都非常简单，而且能为你节省时间。*

---

1. 实例：自动搜集一个网站

要想了解如何自动搜集数据，最好的方法就是用实例来说明。假设你打算下载某个地区去年一整年的温度数据，但你找不到合适的数据来源：要么时间范围不对，要么不是你想要的地区。访问天气网站，一般都只能看到未来10天内的温度预报，而这和你想要的有很大距离。你需要的是以往的实际温度，而不是有关未来的预测。

幸运的是，Weather Underground网站提供了以往的温度。不过你每次只能看到单日的记录。

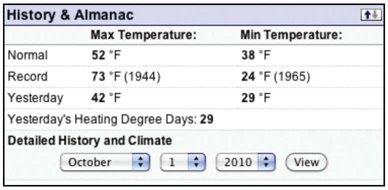
> ▶ Weather Underground的网址是http://wunderground.com。

让我们更具体一些，假设你需要查阅的是纽约州西部港市布法罗。在Weather Underground网站的搜索框里查询“BUF”，会进入布法罗市内尼亚加拉国际机场的天气页面（参见图2-1）。



图2-1 布法罗市的气温，来自Weather Underground网站

页面的顶部提供了该地区当前的温度和其他细节，以及未来5天的天气预报。往下拉到页面的中间会看到History & Almanac（历史年鉴）面板，如图2-2所示。我们可以在下拉菜单中选择某个特定的日期。



History & Almanac

	Max Temperature:	Min Temperature:
Normal	52 °F	38 °F
Record	73 °F (1944)	24 °F (1965)
Yesterday	42 °F	29 °F

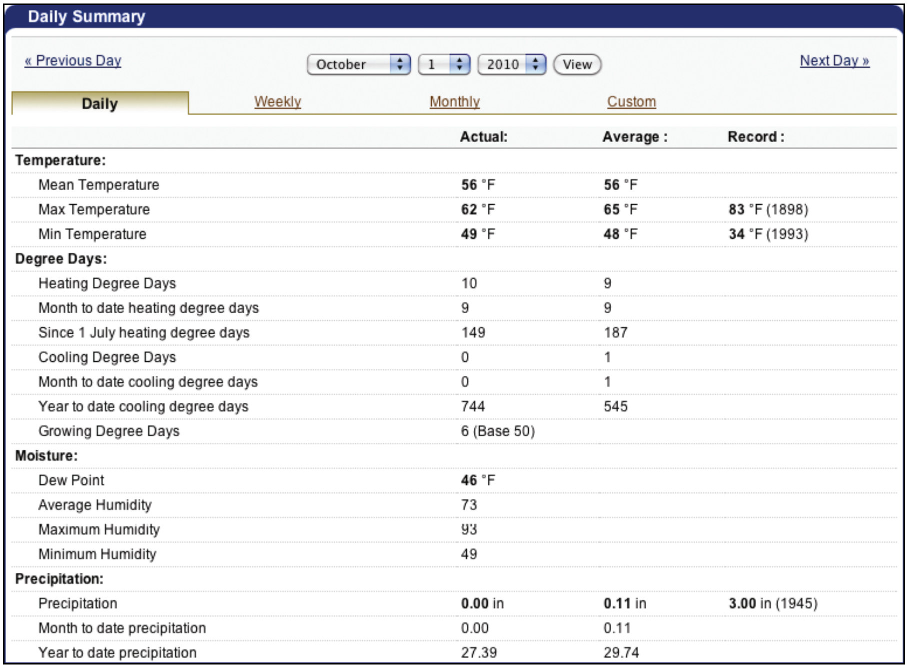
Yesterday's Heating Degree Days: 29

Detailed History and Climate

October 1 2010 View

图2-2 通过下拉菜单选择日期，查看历史数据

在下拉菜单中设置2010年10月1日，单击View按钮。你会看到一个不同的视图，显示了这个日期的细节数据（参见图2-3）。



Daily Summary

« Previous Day　October　1　2010　View　Next Day »

Daily　Weekly　Monthly　Custom

	Actual:	Average :	Record :
Temperature:			
Mean Temperature	**56** °F	**56** °F	
Max Temperature	**62** °F	**65** °F	**83** °F (1898)
Min Temperature	**49** °F	**48** °F	**34** °F (1993)
Degree Days:			
Heating Degree Days	10	9	
Month to date heating degree days	9	9	
Since 1 July heating degree days	149	187	
Cooling Degree Days	0	1	
Month to date cooling degree days	0	1	
Year to date cooling degree days	744	545	
Growing Degree Days	6 (Base 50)		
Moisture:			
Dew Point	**46** °F		
Average Humidity	73		
Maximum Humidity	93		
Minimum Humidity	49		
Precipitation:			
Precipitation	**0.00** in	**0.11** in	**3.00** in (1945)
Month to date precipitation	0.00	0.11	
Year to date precipitation	27.39	29.74	

图2-3　单日的温度数据

页面上提供了温度、日度差、湿度、降雨量等众多数据，但现在你感兴趣的只是每天的最高温度。你可以在第二行第二列找到它。布法罗市在2010年10月1日的最高温度是62°F[①]。

获得这一个数据确实很简单。但要想拿到2009年里每一天的最高温度数值，应该怎么办？最不动脑筋的办法就是不断在下拉菜单那里设置日期。重复365遍，你就大功告成了。

这个过程有意思吗？当然没有。只需要一点点代码和窍门，你就能极快地加速这一过程。为了做到这一点，让我们看看编程语言Python，以及由Leonard Richardson开发的函数库Beautiful Soup。

在下面几段中读者将接触到代码。如果你曾经编过程序，阅读它们就会很轻松。即使你毫无编程经验也不用着急——每一步我都会解释清楚。很多人都把代码视为洪水猛兽、不敢越雷池一步，但请相信我，事情远没有那么可怕。哪怕只掌握一丁点编程技能，都会带来更多的可能性，从而更加自由地处理数据。准备好了吗？让我们开始吧。

首先，请确定你的计算机是否安装有合适的软件。如果你运行的是Mac OS X，那么Python应该已经预先装好。在终端Terminal（应用程序里面）输入“python”启动（参见图2-4）。

---

① 摄氏温度（℃）与华氏温度（°F）之间的转换公式为：℃ =（°F–32）/1.8。——编者注

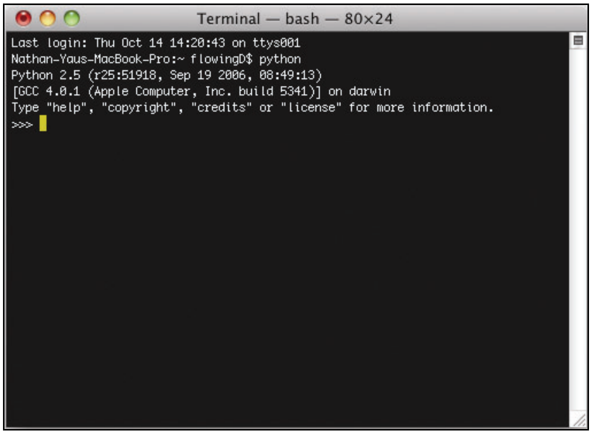


图2-4　在OS X里启动Python

如果你的计算机上安装的是Windows系统，那么可以访问Python网站，根据说明来下载和安装。①

▶访问http://python.org来下载和安装Python。不用紧张，这并不难。

然后，你需要下载Beautiful Soup，它能帮助你方便快速地读取网页。将Beautiful Soup的Python文件（后缀名为.py）保存在你将要保存自己代码的目录中。如果你用过Python，也可以把它放到你自己的库文件路径里面，二者并无区别。

▶访问 www.crummy.com/software/BeautifulSoup/下载Beautiful Soup。下载的版本需要和你所用的Python版本相对应。

安装了Python并下载了Beautiful Soup之后，打开你喜欢的文本或代码编辑器（例如notepad），新建一个文件并存为get-weather-data.py。现在可以开始写代码了。

首先要做的是读取显示历史天气信息的网页。2010年10月1日的布法罗市天气数据的网页URL是：

www.wunderground.com/history/airport/KBUF/2010/10/1/DailyHistory.html?req_city=NA&req_state=NA&req_statename=NA

在这个URL里面，去掉.html后面的东西也一样能访问这个页面，所以让我们把它们去掉。你现在不用关心这些。

www.wunderground.com/history/airport/KBUF/2010/10/1/DailyHistory.html

① 目前Python网站主要提供两个版本的下载：Python 2.x和Python 3.x。不同版本在用途和细节上都存在一些不同。本书作者使用的是Python 2.x版本。

我们发现，URL里面已经通过/2010/10/1标明了日期。由于我们要搜集2009年整年的温度数据，所以把下拉菜单的日期设置为2009年1月1日。现在的URL变成了：

www.wunderground.com/history/airport/KBUF/2009/1/1/DailyHistory.html

所有地方都和10月1日的URL完全一致，只有标识日期的位置例外。现在是/2009/1/1。有意思。那么，如果不通过下拉菜单，怎样才能载入2009年1月2日的页面呢？很明显，只用改动URL里面的日期参数就可以了，它会变成：

www.wunderground.com/history/airport/KBUF/2009/1/2/DailyHistory.html

在浏览器里输入上面这个URL，你就会得到2009年1月2日的天气数据。所以要想获得某一天的天气数据，你要做的就是修改Weather Underground网站的URL。先记住这一点。

现在用Python来读取单个网页，通过下面一行代码先调用`urllib2`函数库：

```
import urllib2
```

要在Python里读取1月1日的网页，调用`urlopen`函数。

```
page = urllib2.urlopen("www.wunderground.com/history/airport/KBUF/2009/1/1/DailyHistory.
html")
```

这个操作会把该URL所指向的页面中的所有HTML都载入进来。下一步就是要从HTML中提取出我们想要的最高温度值。而Beautiful Soup会让我们很轻松地做到这一点。因此在`urllib2`后面我们还需要从Beautiful Soup库中调用BeautifulSoup模块：

```
from BeautifulSoup import BeautifulSoup
```

在文件的最后，用Beautiful Soup来读取（也就是剖析）该网页。

```
soup = BeautifulSoup(page)
```

笼统地说，这行代码是在读取HTML（其本质上也就是一个长字符串），然后以一种易于处理的方式来存储页面上的元素（如标题或图片）。

说明　Beautiful Soup提供了很多说明文档和简单实例。如果对任何地方感到困惑，可以去Beautiful Soup网站进行查阅（就是之前下载函数库的那个网址）。

比如说，如果你想找到该页面上的所有图片，可以这么用：

```
images = soup.findAll('img')
```

这会把Weather Underground页面上所有用`<img />`标签显示的图片都列出来。只想要网页里的第一张图片？那就这么写：

```
first_image = images[0]
```

只想要第二张图片？把0改成1。如果你想要第一个`<img />`标签中的`src`值，那么就这么写：

```
src = first_image['src']
```

好了，我知道你不想要图片。你只想要一个数值：纽约州布法罗市在2009年1月1日的最高温

度。那是26°F。用soup找这个数值比找图片要稍微麻烦一点，但方法是一样的。你只需要确定在`findAll()`里面应该放什么就行了。我们来看一下HTML源文件。

各种主流浏览器都支持这么做。在Firefox里，可以右击页面，选择“查看页面源代码”（Page Source），当前页面的HTML就会在一个新窗口中出现，如图2-5所示。

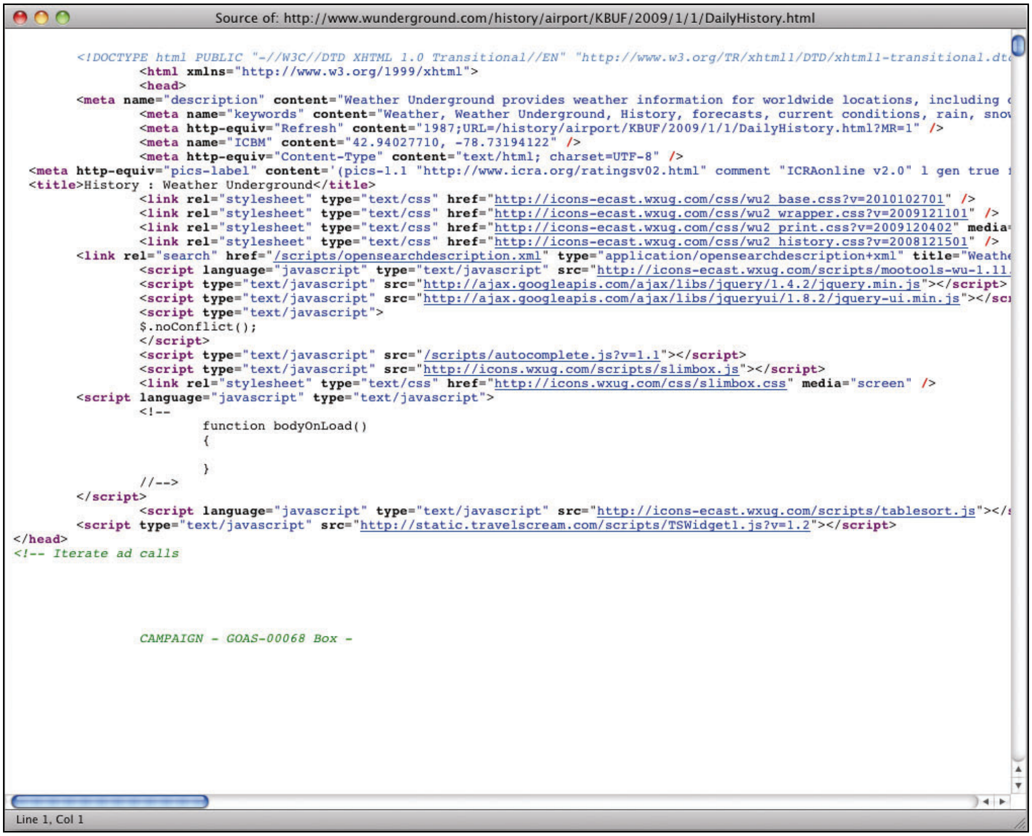


图2-5　Weather Underground页面的HTML源代码

往下拉到显示maximum temperature（最高温度）的地方，或者直接搜索该关键词，这种方法更快。找到26，它就是我们要提取的内容。

这一行有一对封闭的`<span>`标签，其中注明了类名是`nobr`。这就是关键。我们可以找到该页面中所有带有`nobr`类的元素。

```
nobrs = soup.findAll(attrs={"class":"nobr"})
```

和之前一样，只要是带`nobr`类的元素都会被列出来。而我们感兴趣的是其中的第5个，可以通过如下方法找到：

```
print nobrs[4]
```

这个操作带来的是整个元素，但我们只想要26这个数字。在带有nobr类的<span>标签中，还有另一个<span>标签，然后才是26。那么我们应该这样写：

```
dayTemp = nobrs[4].span.string
print dayTemp
```

啊哈！从HTML网页上我们搜集到了第一个数值。下一步就是在2009年的所有页面中进行搜集。要做到这一点，让我们回到最开始的URL。

www.wunderground.com/history/airport/KBUF/2009/1/1/DailyHistory.html

还记得手工修改URL来获得想要的日期吧？之前的代码是针对2009年1月1日的。如果你想要2009年1月2日的网页，只需对URL的日期部分进行相应改动即可。为了获得2009年的所有数据，需要读取每一个月（1～12）以及每个月中的每一天。以下是带有注释的脚本。把它保存到你的get-weather-data.py文件中。

```
import urllib2
from BeautifulSoup import BeautifulSoup


# 创建/打开一个名为wunder-data.txt 的文件（将会是一个以逗号分隔的文本文件）
f = open('wunder-data.txt', 'w')

# 按月和日进行循环访问
for m in range(1, 13):
    for d in range(1, 32):

      # 检查该月是否已经完成
      if (m == 2 and d > 28):
        break
      elif (m in [4, 6, 9, 11] and d > 30):
        break

      # 打开wunderground.com的各个URL
      timestamp = '2009' + str(m) + str(d)
      print "Getting data for " + timestamp
      url = "http://www.wunderground.com/history/airport/KBUF/2009/" +
str(m) + "/" + str(d) + "/DailyHistory.html"
      page = urllib2.urlopen(url)

      # 从页面上获取温度值
      soup = BeautifulSoup(page)
      # dayTemp = soup.body.nobr.b.string
      dayTemp = soup.findAll(attrs={"class":"nobr"})[5].span.string

      # 将月份格式化为时间戳记
      if len(str(m)) < 2:
        mStamp = '0' + str(m)
```

```
        else:
          mStamp = str(m)

        # 将日期格式化为时间戳记
        if len(str(d)) < 2:
          dStamp = '0' + str(d)
        else:
          dStamp = str(d)

        # 创建时间戳记
        timestamp = '2009' + mStamp + dStamp

        # 将时间戳记和温度写入到文本文件中
        f.write(timestamp + ',' + dayTemp + '\n')

# 数据获取结束！关闭文件。
f.close()
```

你应该认得前两行代码，它们调用了必需的库文件urllib2和BeautifulSoup。

```
import urllib2
from BeautifulSoup import BeautifulSoup
```

在此之后，用`open()`函数新建一个名为wunder-data.txt的文本文件，并赋予其写权限。你搜集的所有数据都会被存储到这个文本文件里，和你的脚本文件处于同一个文件目录。

```
# 创建/打开一个名为wunder-data.txt 的文件（将会是一个以逗号分隔的文本文件）
f = open('wunder-data.txt', 'w')
```

下面的代码中使用了一个`for`循环，告诉计算机要读取每个月的数据。月份数字存储在`m`变量中。其后的循环告诉计算机读取每个月中的每一天。日期数字储存在`d`变量中。

```
# 按月和日进行循环访问
for m in range(1, 13):
    for d in range(1, 32):
```

▶查阅Python说明，了解更多循环的工作原理：http://docs.python.org/reference/compound_stmts.html。

注意，我们使用了`range(1, 32)`让程序按天数循环。这表示你可以遍历1～31。然而并不是每一个月都有31天。2月只有28天，而4月、6月、9月和11月都只有30天。没有4月31日的温度数据，因为这一天根本就不存在。所以需要根据月份采取相应的行动。如果当前是2月，在日期超过28的时候就应该中断（break）并前进到下个月。如果你打算搜集多个年份的数据，还需要添加`if`声明以便考虑闰年的情况。

与之类似，如果不是2月而是4月、6月、9月或11月，在当前日期超过30的时候也应该中断

（break）并前进到下个月。

```
# 检查该月是否已经完成
if (m == 2 and d > 28):
  break
elif (m in [4, 6, 9, 11] and d > 30):
  break
```

下面几行代码也应该很眼熟。我们曾通过它们从Weather Underground网站搜集过单个页面的数据。唯一的区别在于URL中的月份和日期变量，这两个地方不应该是静态的。调用urllib2库来载入页面，调用Beautiful Soup来分析页面中的内容，然后通过查找第 5 个`nobr`类来提取最高温度值。

```
      # 打开wunderground.com的各个URL
      url = "http://www.wunderground.com/history/airport/KBUF/2009/" +
str(m) + "/" + str(d) + "/DailyHistory.html"
      page = urllib2.urlopen(url)

      # 从页面上获取温度值
      soup = BeautifulSoup(page)
      # dayTemp = soup.body.nobr.b.string
      dayTemp = soup.findAll(attrs={"class":"nobr"})[5].span.string
```

剩下的一块代码将根据年份、月份和日期来组成时间戳记。时间戳记的格式是yyyymmdd。当然，任何其他格式也都没问题。

```
# 将月份格式化为时间戳记
if len(str(m)) < 2:
  mStamp = '0' + str(m)
else:
  mStamp = str(m)

# 将日期格式化为时间戳记
if len(str(d)) < 2:
  dStamp = '0' + str(d)
else:
  dStamp = str(d)

# 创建时间戳记
timestamp = '2009' + mStamp + dStamp
```

最后，温度值和时间戳记都会通过`write()`函数写入到wunder-data.txt文件中去。

```
# 将时间戳记和温度写入到文本文件中
f.write(timestamp + ',' + dayTemp + '\n')
```

在所有月份和日期都获取完成后，使用`close()`来关闭wunder-data.txt文件。

```
# 数据获取结束！关闭文件
f.close()
```

剩下的事情就是运行代码了，在终端输入：

```
$ python get-weather-data.py
```

运行需要一段时间，请保持耐心。计算机在这个过程中实际上读取了365个网页，以对应2009年的每一天。在脚本运行结束后在你的工作路径下会出现一个名为wunder-data.txt的文件。打开它就会看到你想要的数据，以逗号分隔。第一列是时间戳记，第二列是温度值。看上去应该类似图2-6中的样子。

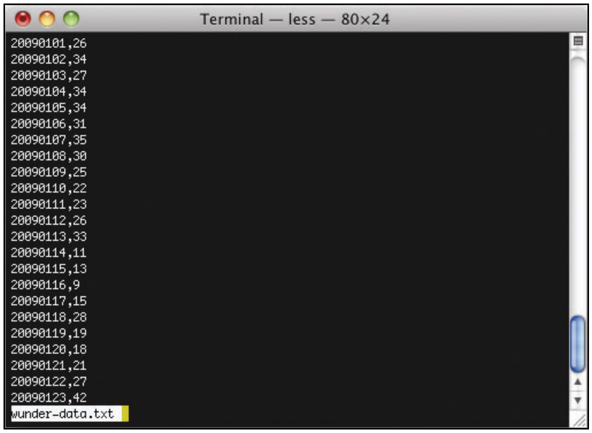


图2-6　搜集到的整整一年的温度数据

2. 案例归纳

我们从Weather Underground网站上搜集到了天气数据，而对于其他数据来源，也可以借鉴这个过程。典型的自动搜集数据包括三个步骤：

(1) 找出规律；

(2) 循环；

(3) 存储数据。

在上面这个例子中，我们必须找到两个规律。第一个在URL里面，第二个则是在网页中找到具体的温度值。要想载入2009年不同日期的网页，需要改动URL中的月份和日期部分。而温度值则位于HTML中的第5个`nobr`类标签中。如果在URL中没有明显的规律，可以换个角度想想怎样得到所有这些页面的URL。也许可以看一下网站地图，或者通过搜索引擎查阅网站目录。无论如何，你必须知道所有数据页面的URL。

在找到规律之后，我们要进行循环处理。也就是说，需要通过程序访问所有的页面，载入这些页面，然后读取并分析。例子中我们用到的是Beautiful Soup，它能让Python更容易地分析XML和HTML。如果你选择的是其他编程语言，也应该有类似的函数库。

最后，我们需要把数据存储在某个地方。最简单的办法是把数据存储为纯文本文件，以逗号作为分隔符，但如果你有自己的数据库，也可以存到那里面。

有些网页中显示的数据是通过JavaScript来调用的，这种情况就会麻烦一些，但整个搜集的过程还是一样。

2.2 设置数据的格式

不同的可视化工具支持不同的数据格式，而且所用到的结构也会根据你打算讲的故事而有所差异。所以，你的数据结构越灵活，带来的可能性也会越多。运用数据格式化工具，再加上一点编程技巧，你就能为数据安排各种不同的格式、满足各种需求。

最简单的办法当然是找一个程序员来帮你做数据的分析和格式化工作，但你总是得看人脸色。这一点在任何项目的早期阶段尤为明显，在此阶段，循环处理和数据探索对于可视化设计来说都是非常关键的。说句实话，如果我是用人部门，我也喜欢招收那些了解如何处理数据的人，而不是在每个项目初期都需要别人帮助的人。

我对格式化的理解

当我在高中接触统计学时，需要的数据都有着清楚的、矩阵式的格式。我要做的就是把其中一些数字导入到Excel表格或者当时最时髦的图形计算器中去（上课时用这玩意看似在认真计算什么，其实大多时候都在玩俄罗斯方块）。到大学后也是如此，因为我学习的是数据分析理论与技巧，老师们不会在原始的、未处理过的数据上浪费时间。数据一直都以合适的格式呈现。

考虑到当时的学习重点和课业进度限制，这很容易理解。但开始研究生课程之后，我意识到真实世界里的数据从来都不会按自己需要的格式呈现。你会遇到各种情况，例如某个数值莫名其妙的丢失、标记前后矛盾、没有任何上下文背景，或者千奇百怪的笔误。数据通常都散落在多个表格中，但我们需要的是一个包含所有数据的表格，而且带有各自的名称或独立的ID号。

在我开始可视化工作之后也是如此。而且格式化变得越来越重要，因为我希望用手中的数据做更多的事情。如今，我在数据格式化上花的精力和可视化一样多，有时候甚至超过了可视化的时间。乍一听这似乎有点奇怪，但你会发现当所有的数据都经过精心组织后，对图形的设计会变得更加简单，就像回到了高中的入门课程一样。

下面将介绍各种数据格式、处理格式的工具以及一些编程知识。编程中涉及的逻辑思路和之前搜集数据时所用的几乎相同。

2.2.1 数据格式

大多数人都习惯用Excel来处理数据。如果你打算从分析到可视化一直都用这款软件，那么没问题。但如果你想跨出这个圈子，就需要熟悉一下其他数据格式。这些格式之所以存在，是为了让你的数据机器可读，也就是设置成计算机能够理解的格式。使用何种格式取决于你的意图和所用的可视化工具，不过以下3种格式基本上可以满足所有需求：带分隔符的文本、JavaScript对象表示法（JavaScript Object Notation）和可扩展标记语言（eXtensible Markup Language）。

1. 带分隔符的文本

很多人都很熟悉带分隔符的文本。我们在前面一节的例子中就创建过以逗号分隔的文本文件。如果把数据集看成是按行和列来分布，那么分隔符文本就是用分隔符来分开每一列。分隔符一般用的是英文逗号（半角字符），也可以是制表符tab，或者是空格、英文分号、冒号、斜杠等任何你喜欢的字符。不过逗号和tab是最常见的。

分隔符文本应用广泛，可以被大多数电子表格程序阅读，例如Excel或者Google Documents。我们也可以把电子表格输出成分隔符文本。如果你要使用多个工作表格，通常就会有多个分隔符文件，除非特殊指定。

这种格式也便于与其他人共享，因为它无需依赖于任何特定程序。

2. JavaScript对象表示法（JSON）

很多网页API都适用于这种格式。它既能够让计算机理解，又便于人类阅读。不过如果你眼前的数据过多，盯太久可能会头晕目眩。该格式基于JavaScript表示法，但并不依赖于这种语言。JSON中有许多规格说明，但只用掌握一些基础就能满足大部分需要。

JSON利用关键字和值，并且把数据条目作为对象来处理。如果我们把JSON数据转化成逗号分隔数据（Comma-Separated Value，CSV），那么每个对象都会占一行。

大家将会在后文中看到，有很多应用、语言和函数库都支持JSON输入。如果你打算设计便于互联网传播的数据图形，就得了解一下这种格式。

> ▶访问http://json.org阅读JSON的完整说明。你不必了解这一格式的所有细节，但当你需要使用某个JSON数据源时，它还是很管用的。

3. XML

XML（可扩展标记语言）是另一种互联网上的流行格式，常被用于在API间传递数据。XML分为很多类型，规格说明也不少，但从最基本的层面来看，它就是一个文本文件，其中的值都封闭在各种标签之内。比如说，人们用于订阅各种博客RSS（如FlowingData）的feed就是一个XML文件，如图2-7所示。

目前的RSS列表中，数据条目都被封闭在`<item></item>`标签中，每一条都带有各自的标题、描述、作者、发布日期以及其他属性。

XML相对比较容易用函数库来分析，例如Python里面的Beautiful Soup。在本书后面的内容中，大家会更加熟悉XML、CSV以及JSON。

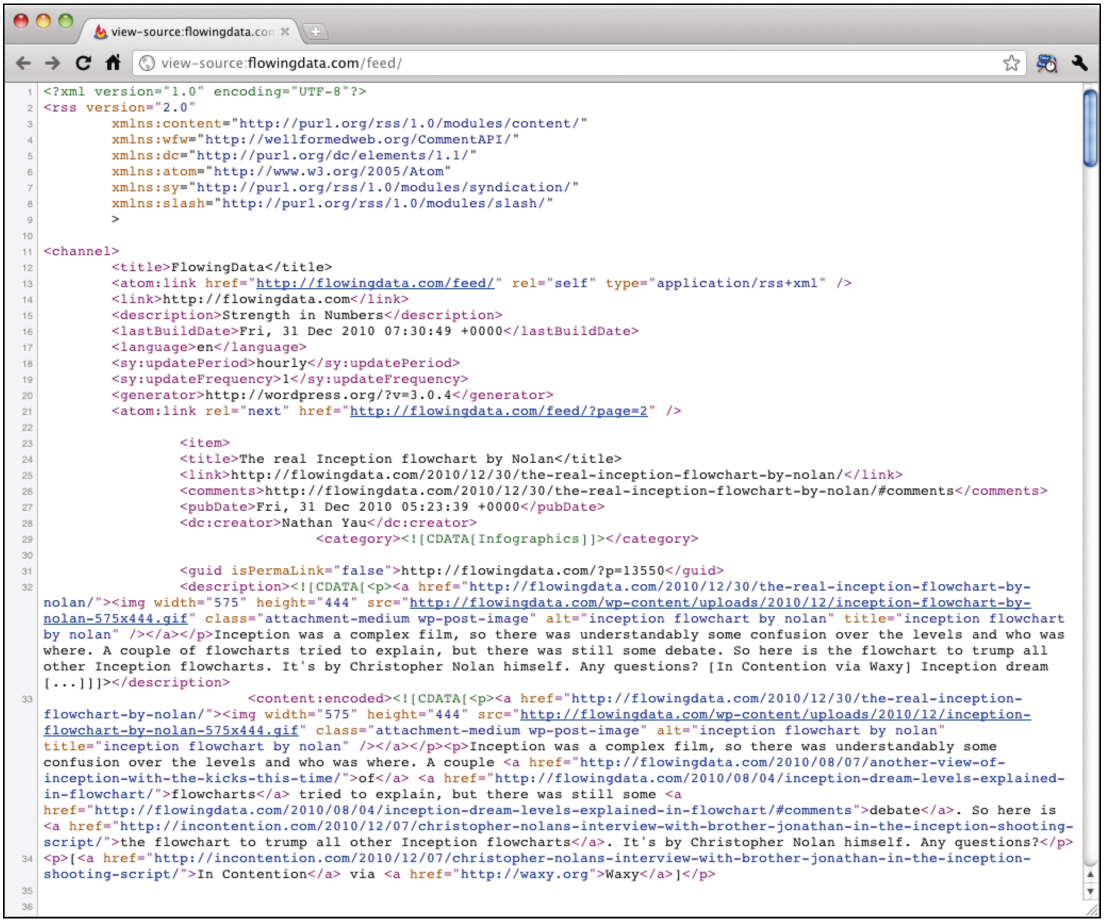


图2-7　FlowingData网站RSS的feed片段

2.2.2　格式化工具

在几年前，人们需要写一些小脚本才能对数据进行处理和格式化。在写了几次脚本后，你就会注意到其中的逻辑规律，所以即使每逢新的数据集都需要写新的代码，也并不是什么难事。不过每次都这样做也确实挺花时间。幸而随着数据处理量的逐渐增大，有人开发了一些工具来代替我们进行这种千篇一律的手工劳动。

1. Google Refine

Google Refine是Freebase Gridworks的进化版本。Gridworks最初是Freebase这个开放数据平台的内部工具，但是后来Freebase被Google收购，所以Gridworks也就换了名字。从本质上来说，Google Refine就是易用性更强、功能更多的Gridworks 2.0版本（参见图2-8）。

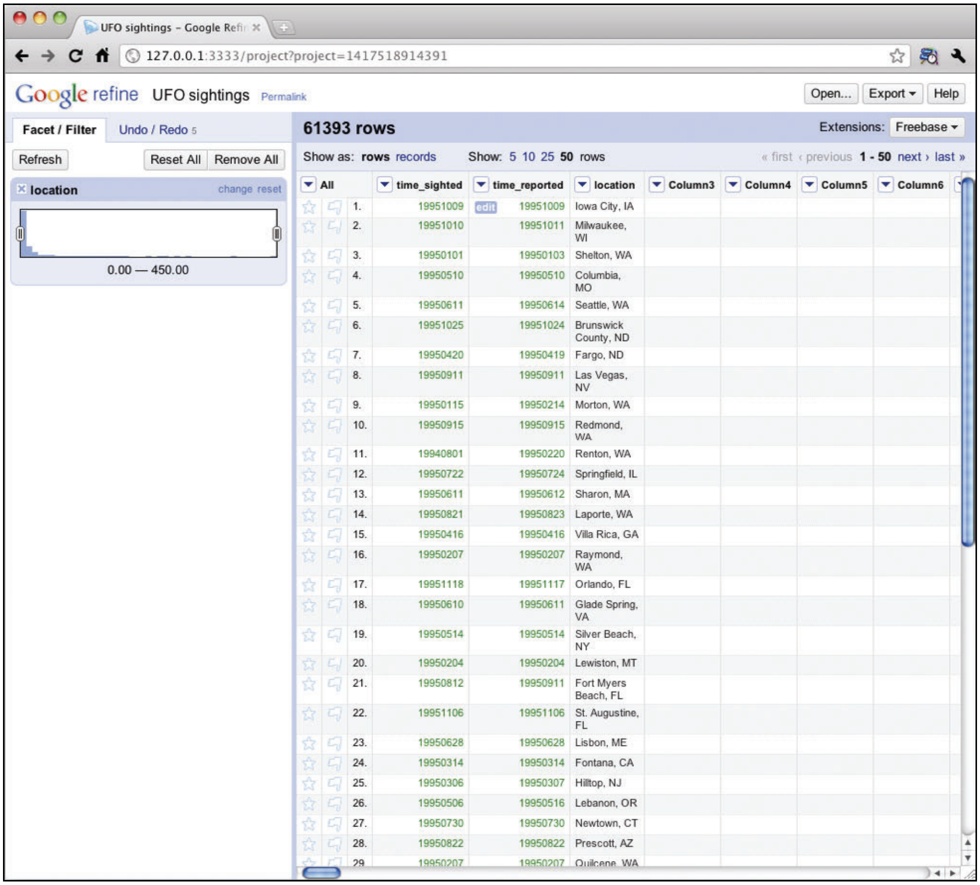


图2-8　Google Refine的用户界面

它会在你的桌面运行（但依然会通过浏览器），这一点很棒，因为我们无需担心私密数据会上传到Google服务器去，所有的处理都在自己的计算机上进行。Refine也是开源的，如果你有需求，也可以将它扩展到你自己的应用中去。

在运行Refine时，我们会看到一个熟悉的电子表格界面。数据都按照行和列的形式呈现，便于排序或者搜索某个值，同时也能很轻松地找出数据中的矛盾之处从而加以改进。

比如说，你需要建立一个自己家厨房的财产清单。在Refine中载入数据后很快就能发现笔误、分类不一致等前后矛盾的地方。也许fork（叉子）无意中被拼成了“frk”，或者你突然想到应该把所有的叉子、勺子和餐刀全部归为“餐具”类中。在Refine里面这些都很容易办到。如果你不喜欢刚刚作出的改动，用undo（撤销）命令就能让数据集恢复原状。

要想再深入一步，你还可以导入其他数据来源（比如Freebase里的数据集），从而创建一个更大的数据集。

Google Refine是一款很好的工具，它功能强大，而且提供免费下载。强烈建议大家至少一试。

▶访问http://code.google.com/p/google-refine/，下载开源的Google Refine并浏览教程，学习如何最大限度地利用这款工具。

2. Mr. Data Converter

我们常常会发现，所有的数据都在Excel里面，但却需要把它们转换成其他格式。在为互联网创建数据图时这种情况尤其常见。你可能已经把Excel电子表格输出为CSV了，但如果需要其他格式该怎么办？Mr. Data Converter可以帮助你。

Mr. Data Converter是一款简单、免费的工具，创造者是Shan Carter，《纽约时报》的一名图形编辑。Carter的主要工作是为报纸的在线版本创建交互式数据图，所以他经常得把数据转换成他的软件所支持的格式。难怪他制作了这样一款工具。

图2-9显示了这款工具的界面，非常简单易用。你所要做的只是把Excel里面的数据复制粘贴到页面上方的文本框里面，然后选择想要的输入格式即可。有XML、JSON以及其他多种格式可选。

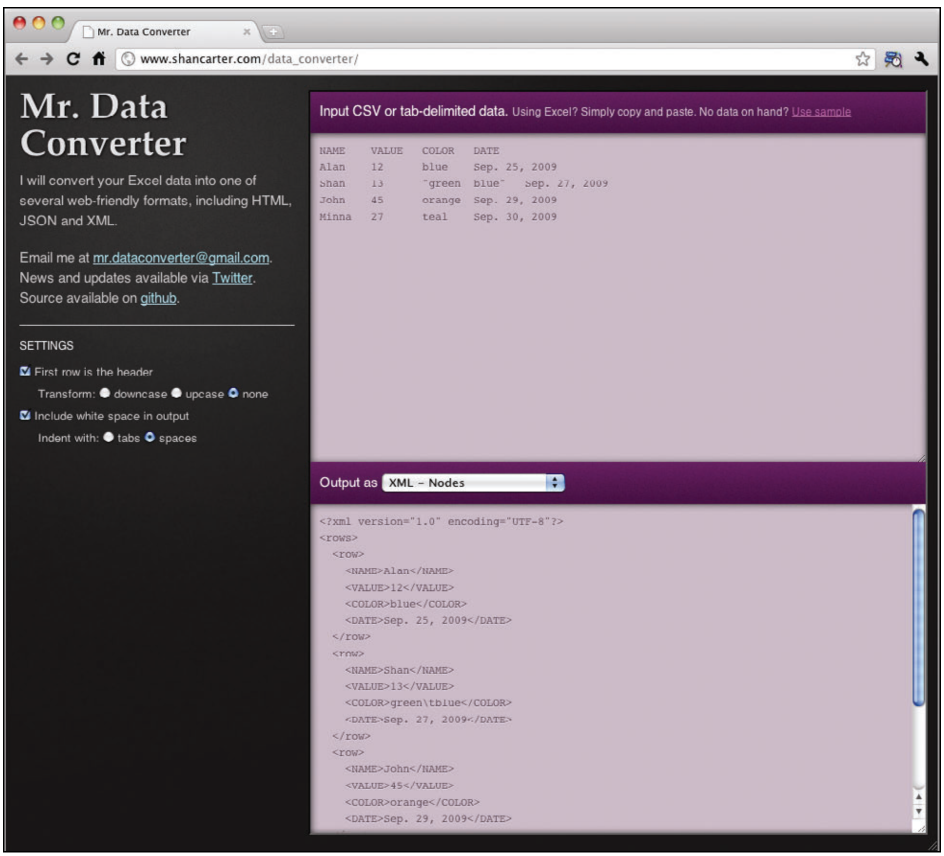


图2-9　Mr. Data Converter让数据格式之间的转换变得更加容易

如果你想对Mr. Data Converter加以扩展，它的源代码同样也是开放的。

▶访问http://www.shancarter.com/data_converter/试用Mr. Data Converter，将你的Excel电子表格转换为Web友好的格式。或者到github下载源代码，地址是https://github.com/shancarter/Mr-Data-Converter。

3. Mr. People

受到Mr. Data Converter的启发，《纽约时报》的图形副主编Matthew Ericson创建了Mr. People。和Mr. Data Converter一样，Mr. People也允许在文本框中复制、粘贴数据，然后进行分析和提取工作。不过正如Mr. People这个名字所暗示的那样，它主要用于分析姓名。

也许你手里有一长列姓名数据，你想识别出其中的每一个姓氏和名字（包括中间名缩写、前缀和后缀），但是它们的格式却很混乱，很多姓名都被列在同一行中。这就是Mr. People能一显身手的地方。把所有的名字都复制粘贴到Mr. People里（参见图2-10），你就会得到一个非常干净的表格，可以复制到其他电子表格软件中去，如图2-11所示。

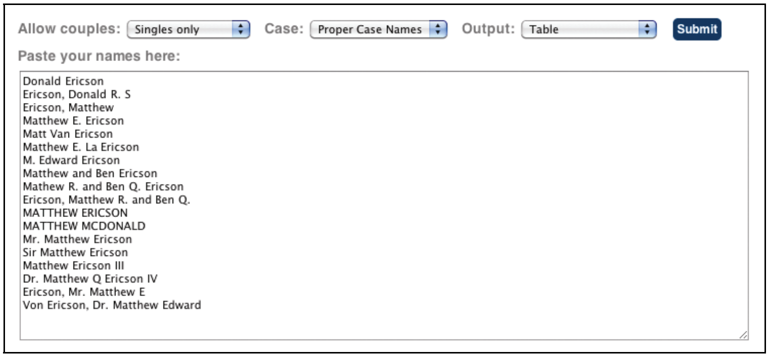


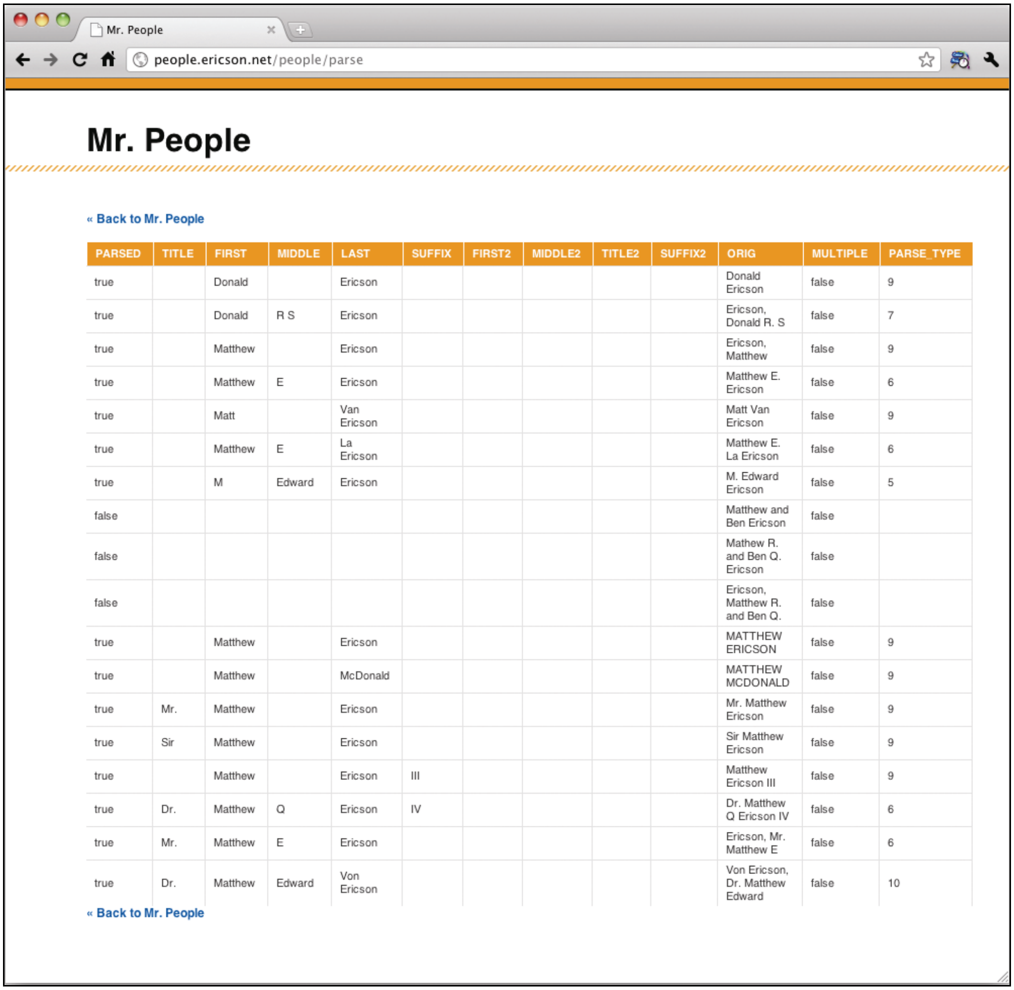
图2-10 Mr. People的姓名输入界面

和Mr. Data Converter一样，Mr. People也是一款开源软件，可在github上下载。

▶访问http://people.ericson.net/使用Mr. People来对姓名进行分析，或者到http://github.com/mericson/people下载Ruby源文件，用在你自己的脚本里。

4. 电子表格软件

当然了，如果你只是想排序，或者只是对某个数据点进行一些小改动，那么你还是可以用自己喜欢的那些电子表格软件。如果你不排斥手工编辑数据，这样做完全没问题。其他情况下，还是用之前提到的那些工具为好（尤其是当数据集非常庞大时），或者自己编程来解决。



PARSED	TITLE	FIRST	MIDDLE	LAST	SUFFIX	FIRST2	MIDDLE2	TITLE2	SUFFIX2	ORIG	MULTIPLE	PARSE_TYPE
true		Donald		Ericson						Donald Ericson	false	9
true		Donald	R S	Ericson						Ericson, Donald R. S	false	7
true		Matthew		Ericson						Ericson, Matthew	false	9
true		Matthew	E	Ericson						Matthew E. Ericson	false	6
true		Matt		Van Ericson						Matt Van Ericson	false	9
true		Matthew	E	La Ericson						Matthew E. La Ericson	false	6
true		M	Edward	Ericson						M. Edward Ericson	false	5
false										Matthew and Ben Ericson	false	
false										Mathew R. and Ben Q. Ericson	false	
false										Ericson, Matthew R. and Ben Q.	false	
true		Matthew		Ericson						MATTHEW ERICSON	false	9
true		Matthew		McDonald						MATTHEW MCDONALD	false	9
true	Mr.	Matthew		Ericson						Mr. Matthew Ericson	false	9
true	Sir	Matthew		Ericson						Sir Matthew Ericson	false	9
true		Matthew		Ericson	III					Matthew Ericson III	false	9
true	Dr.	Matthew	Q	Ericson	IV					Dr. Matthew Q Ericson IV	false	6
true	Mr.	Matthew	E	Ericson						Ericson, Mr. Matthew E	false	6
true	Dr.	Matthew	Edward	Von Ericson						Von Ericson, Dr. Matthew Edward	false	10

图2-11　经过Mr. People分析后得到的姓名表格

2.2.3　用代码来格式化

虽然那些操作方便的软件确实能够带来帮助，但有时它们用起来还是谈不上得心应手。而且有些软件不善于应付大型数据，它们会变慢或者崩溃。

这种情况下你该怎么办呢？你可以放弃，但恐怕也于事无补。不过，你可以写一点代码来搞定这件事。有了代码的帮助，处理起来就会更加游刃有余，而且还能随时因地制宜地修改脚本。

现在让我们用一个实例来说明如何用几行代码实现数据格式之间的切换。

1. 实例：数据格式的切换

本例中使用的是Python，不过你也可以用其他任何喜欢的语言。所用的思路是一样的，但语

法上可能会有所不同。（我喜欢用Python来开发应用，因此自然会选择用Python来处理原始数据。）

让我们接着用之前数据搜集时得到的wunder-data.txt文件，其中有纽约州布法罗市2009年的温度数据。最初的几行应该是这样：

```
20090101,26
20090102,34
20090103,27
20090104,34
20090105,34
20090106,31
20090107,35
20090108,30
20090109,25
...
```

这是个CSV文件，如果你希望数据是以下的XML格式：

```
<weather_data>
    <observation>
        <date>20090101</date>
        <max_temperature>26</max_temperature>
    </observation>
    <observation>
        <date>20090102</date>
        <max_temperature>34</max_temperature>
    </observation>
    <observation>
        <date>20090103</date>
        <max_temperature>27</max_temperature>
    </observation>
    <observation>
        <date>20090104</date>
        <max_temperature>34</max_temperature>
    </observation>
    ...
</weather_data>
```

其中每一天的温度数据都位于`<observation>`标签内，包括日期`<date>`和最高温`<max_temperature>`。

要想把CSV文件转换为这种XML格式，可以使用以下代码片段：

```
import csv
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")
print '<weather_data>'

for row in reader:
    print '<observation>'
```

```
    print '<date>' + row[0] + '</date>'
    print '<max_temperature>' + row[1] + '</max_temperature>'
    print '</observation>'

print '</weather_data>'
```

和之前一样，首先你需要导入必要的模块。因为本例中读取的是CSV文件wunder-data.txt，所以我们只需要导入csv模块即可。

```
import csv
```

第2行代码先通过open()函数打开wunder-data.txt，然后通过csv.reader()函数对其进行读取。

```
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")
```

请注意分隔符被指定为一个逗号。如果文件是以制表符tab分隔，就需要指定分隔符为'\t'。

第3行代码的作用是输出XML文件的开篇第一行。

```
print '<weather_data>
```

在代码的主干部分，你可以在CSV文件中循环读取每一行数据，并输出所需要的XML。在本例中，CSV文件的每一行都对应XML中的一个<observation>标签。

```
for row in reader:
    print '<observation>
    print '<date>' + row[0] + '</date>'
    print '<max_temperature>' + row[1] + '</max_temperature>
    print '</observation>'
```

每一行都有两个值：日期和最高温度。

最后将标签封闭，结束XML转换。

```
print '</weather_data>
```

这段代码主要完成两项任务。首先是读取数据，其次是循环处理，修改每一行数据的格式。如果想把生成的XML转换回CSV，整个思路也是一样的。正如以下代码片段所示，唯一的区别是需要另一个模块来分析XML文件。

```
from BeautifulSoup import BeautifulStoneSoup

f = open('wunder-data.xml', 'r')
xml = f.read()

soup = BeautifulStoneSoup(xml)
observations = soup.findAll('observation')
for o in observations:
    print o.date.string + "," + o.max_temperature.string
```

这段代码看上去跟前面很不一样，但本质上做的是同一件事。我们没有再用csv模块，而是

从BeautifulSoup库中输入了BeautifulStoneSoup模块。还记得我们曾用过BeautifulSoup模块来解析Weather Underground 网站的HTML吧？BeautifulStoneSoup则用于解析更为广泛的XML。

通过`open()`函数打开XML文件，然后载入`xml`变量中的内容。此时`xml`内容会被存储为字符串的形式并传递到`BeautifulStoneSoup`。通过`findAll()`函数获取所有的`<observation>`标签并进行循环分析。最后，和我们将CSV转换为XML的过程一样，循环所有的`<observation>`，将其中的值输出成我们需要的格式。

2

这样我们就得到了最开始的CSV文件：

```
20090101,26
20090102,34
20090103,27
20090104,34
...
```

为了加深大家的认识，下面提供了将CSV转换为JSON格式的代码。

```
import csv
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")

print "{ observations: ["
rows_so_far = 0
for row in reader:

    rows_so_far += 1

    print '{'
    print '"date": ' + '"' + row[0] + '", '
    print '"temperature": ' + row[1]

    if rows_so_far < 365:
        print " },"
    else:
        print " }"

print "] }"
```

你可以逐行阅读上面的代码，试着理解它们的意思。其实和之前一样，虽然输出的结果不同，但整个思路是一致的。以下是运行上面代码得到的JSON文件片段。

```
{
    "observations": [
        {
            "date": "20090101",
            "temperature": 26
        },
        {
            "date": "20090102",
```

```
            "temperature": 34
        },
        ...
    ]
}
```

数据还是那些数据，但是日期和温度却变成了另一种格式。计算机就是喜欢多样性。

2. 在循环中加入新的逻辑

回顾一下将CSV文件转换为JSON时所用的代码，你应该会注意到在`for`循环中的三行`print`代码之后使用了`if-else`语句。它的作用是检查当前迭代是否已经到达最后一行数据。如果没有到达，该条记录的末尾就加上一个逗号；如果已经到达，则不加逗号。这是JSON规范中的一部分。实际上，在这里我们可以做更多事情。

我们可以检查最高温度是否大于某个临界值，比如创建一个新的值：1代表超过该临界值，0代表未超过。此举可以让我们对温度数据进行分类，或者标记出某些特殊的日子。

当然，我们能做的并不仅限于检查临界值。比如说可以计算某段时间内的流动平均温度，或者每天与前一天的温度差。要想拓展原始数据的用途，利用循环几乎可以随心所欲，不过先让我们看一个简单的例子。

让我们回到最初的CSV文件wunder-data.txt，通过代码创建第三列，指出当天的最高温度是高于还是低于冰点。0表示温度高于冰点（32°F以上），而1表示温度等于或低于冰点。

```
import csv
reader = csv.reader(open('wunder-data.txt', 'r'), delimiter=",")
for row in reader:
    if int(row[1]) <= 32:
        is_freezing = '1'
    else:
        is_freezing = '0'

    print row[0] + "," + row[1] + "," + is_freezing
```

运行代码后，结果会直接显示在终端里。和之前一样，程序将CSV文件里的数据读取到Python中，然后循环处理每一行，检查对应的每一天并作标记。

这当然是一个很简单的例子，但从中不难发现，我们可以对这一逻辑进行扩展，从而随心所欲地拓展或格式化数据。只需记住载入、循环和处理这三个步骤即可。

2.3　小结

本章讲述了怎样寻找需要的数据，以及在获得数据后应如何处理。在可视化过程中，这是至关重要的一步。如果基础数据让人感觉乏味，数据图也不可能会让人产生兴趣。无论图形设计得多么优美，只有数据（或者分析数据得到的结果）本身才是立足之本。现在大家掌握了获取和处理数据的方法，已经迈出了很大一步。

与此同时我们也初步接触了编程。从网站搜集数据、赋予其格式并重新整理，这些技巧将为后面几章的学习带来很大帮助。不过，最重要的一点在于代码中的逻辑思路。虽然我们只用到了Python，但在此基础上使用Ruby、Perl或者PHP都不是什么难事。各种编程语言的思路都是一致的。当你掌握了一门编程语言之后，学习其他语言会非常容易（如果你对编程很了解的话，那么一定会赞同这句话）。

当然，不一定每次都非得求助于代码。如果有方便的应用能够帮你分担一部分工作，那么就应该尽量利用它们。毕竟你掌握和了解的工具越多，卡壳的概率就会越小。

很好，你手上已经有了数据。可以开始可视化了。

选择可视化工具

3

在上一章中，我们了解了去哪里找到数据，以及如何将数据设置为自己需要的格式。现在让我们看看可视化方面。关于这一点，人们最常问的问题是“我应该用什么软件来对数据可视化？”

幸运的是，我们有很多选择。有些软件是开箱即用的（out-of- the-box），只需要鼠标就能操作，还有些软件则需要一点编程技巧。虽然有些工具并非专门用于制作数据图，但依然很有帮助。它们在本章中都会有所提及。

掌握或了解的可视化工具多了之后，面对数据时就不会束手无策，创作出你想要的数据图也会更加容易。

3.1 开箱即用的可视化工具

开箱即用的软件是目前最简单的解决方案，适合新手学习。只需对数据进行一些复制粘贴，或者载入某个CSV文件就可以开始了。直接选择想要的图形类型，然后稍微调整一下选项即可。

3.1.1 可选项

开箱即用的工具彼此之间有很多不同，这取决于它们各自面向的对象。其中一些（例如Microsoft Excel和Google Documents）主要用于基础的数据管理和图形创建，而另一些则更偏向于深入分析或可视化研究。

1. Microsoft Excel

几乎所有人都知道这款软件。图3-1显示了输入数据时的电子表格。

输入一定数据之后，在菜单栏单击“图表”的选项就可以生成想要的图表了。Excel提供了各种标准的图表类型以供选择，包括柱形图、折线图、饼图和散点图（scatter plot）等，如图3-2所示。

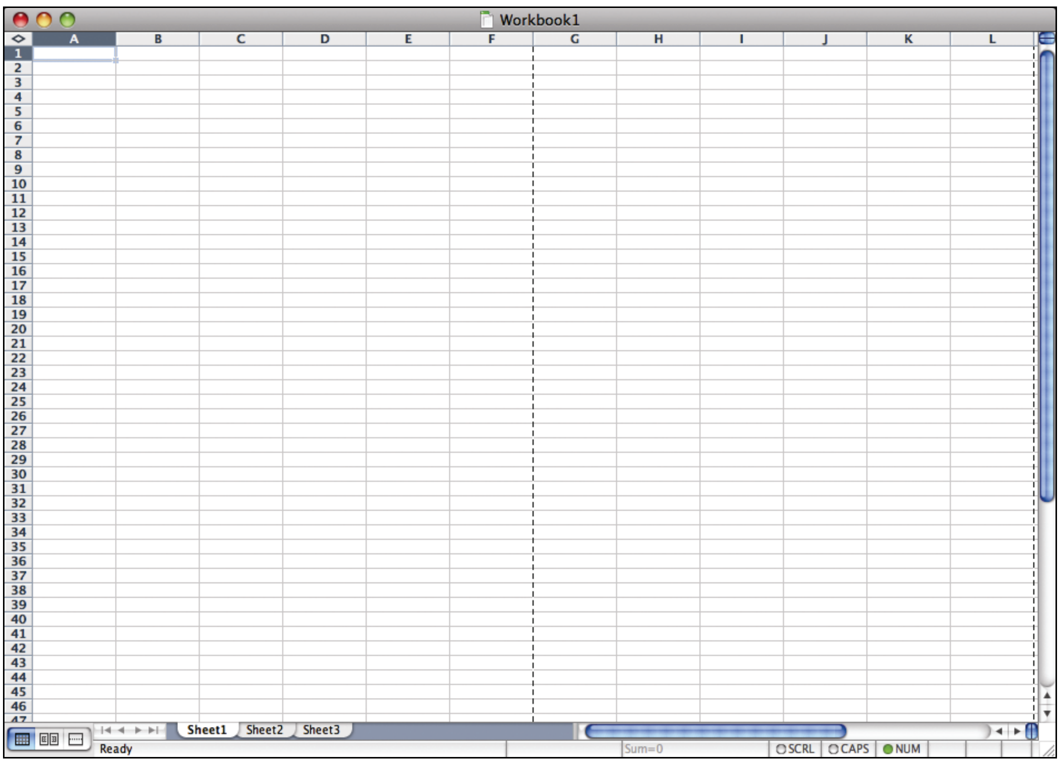


图3-1 Microsoft Excel的电子表格

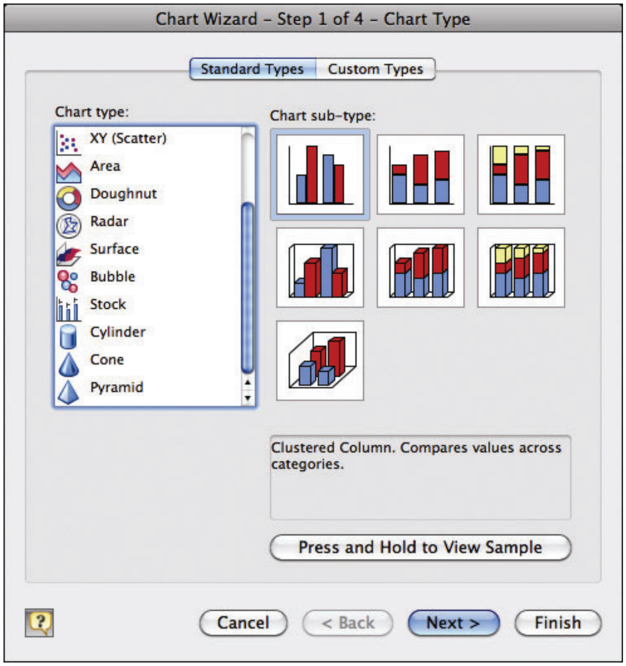


图3-2 Microsoft Excel的图表选择

有些人对Excel嗤之以鼻，但其实它并不至于如此不堪。比如说，虽然我不会用Excel来做深度分析，生成的图表也不会用于出版，但如果我手中正好有一个Excel格式的小型数据集（这种情况很常见），同时又想快速找找感觉，那么自然就会随手点击几下鼠标用它生成一个图形。

图形也可以很好玩

我在计算机上制作的第一张数据图就是用的Microsoft Excel，是为了应付小学四年级的一次课外科学研究。我和搭档试图研究蜗牛在什么样的平面上爬得最快。这是一次史无前例的研究，我向你保证。

虽然还是个小屁孩，那时候我就已经开始喜欢上制图了。我花了很长时间才学会（当时计算机对我来说还是个新鲜玩意），但在最终掌握之后，回报是令人激动的。我在电子表格里输入数字，然后立刻就能得到图形，还可以随意改变颜色，一切都棒极了。

正是Excel的方便易用让它获得了大众的欢迎，这很好。但如果你想要高质量的数据图就不要止步于此。其他工具会更适合你。

2. Google Spreadsheets

Google Spreadsheets其实就是Microsoft Excel的云版本，两者的界面非常相似（参见图3-3）。

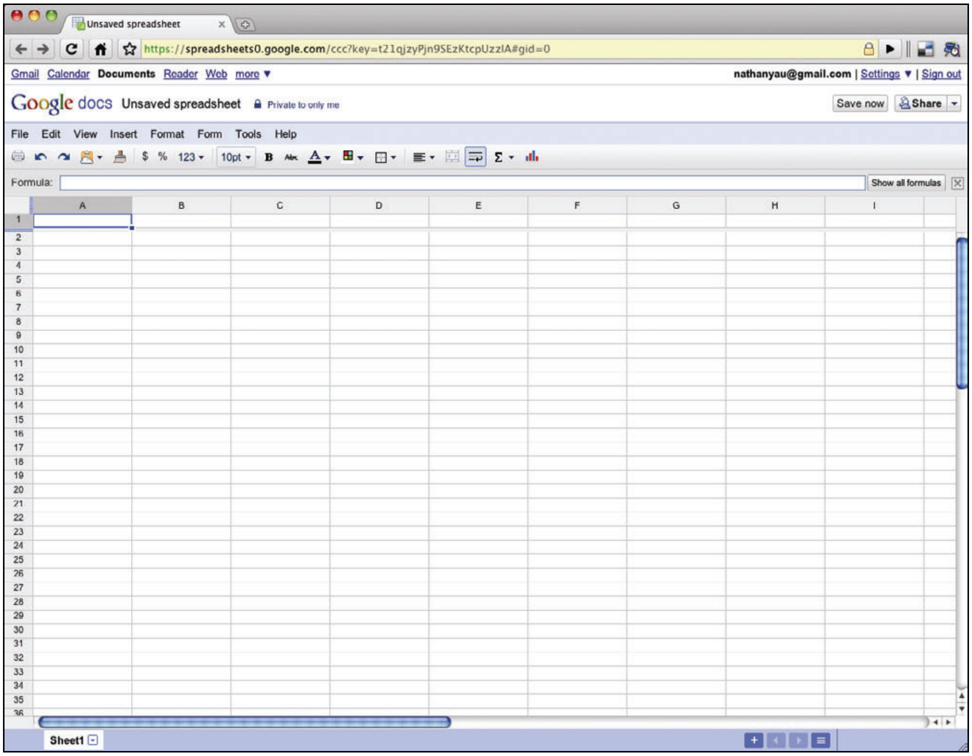


图3-3　Google Spreadsheets

它也提供了标准的图表类型，如图3-4所示。

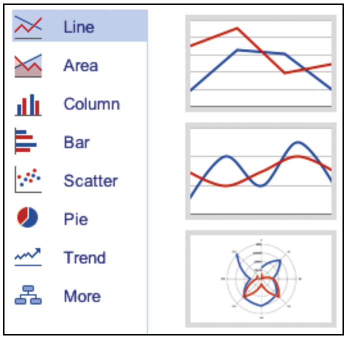


图3-4　Google Spreadsheets的图表选择

不过和Excel相比，Google Spreadsheets拥有更多优势。首先，你的数据都存储在Google的服务器上，所以你可以在任何安装了浏览器的计算机上查看自己的数据，只要登录你的Google账号就行。同时你也能很方便地把自己的电子表格共享给其他人，还能做到实时协作。此外，Google Spreadsheets还在Gadget（小工具）选项中提供了很多其他的图表类型，如图3-5所示。

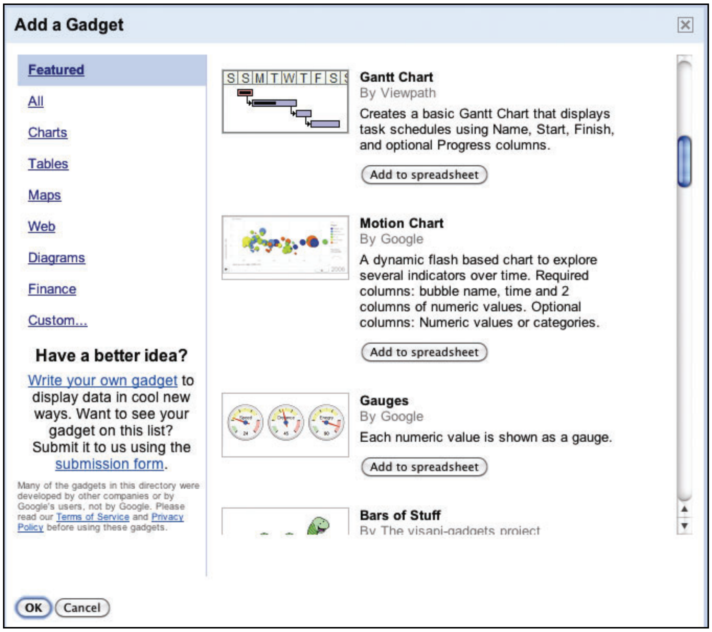


图3-5　Google小工具

大多数小工具都没什么用，但其中也有几个好的。比如说，你可以轻而易举地为自己的时间序列数据创建运动图表（就像Hans Rosling做的那样）。此外还有一种可交互的时间序列图表，如果你访问过Google Finance的话，就会发现它非常眼熟（参见图3-6）。



图3-6　Google Finance

▶访问Google Docs试用一下spreadsheets，地址是http://docs.google.com。

3. Many Eyes

Many Eyes是IBM视觉传达实验室（IBM Visual Communication Lab）主导的一个研究项目，目前还在进行中。它是一个在线应用，带有一系列交互式的可视化工具，可以识别带分隔符的文本文件。Many Eyes的初衷是想了解人们能否以群组的形式探索大型数据集——这也正是它的名字的来历。如果一个群组内的众多双眼睛来观察某个数据集，是否会从中挖掘到更多有意思的地方？效率是否会更高？

虽然Many Eyes目前尚未提供多人的数据分析功能，作为个人使用来说它依然很有价值。其中涵盖了绝大多数传统的可视化类型，例如折线图（图3-7）和散点图（图3-8）。

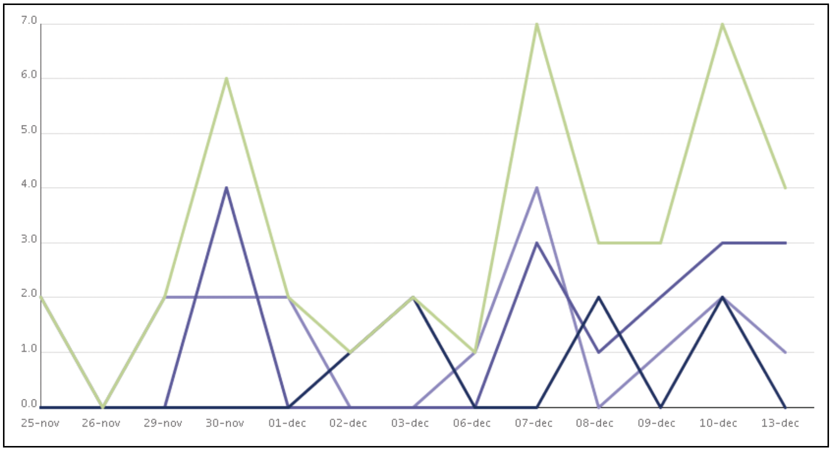


图3-7　Many Eyes的折线图

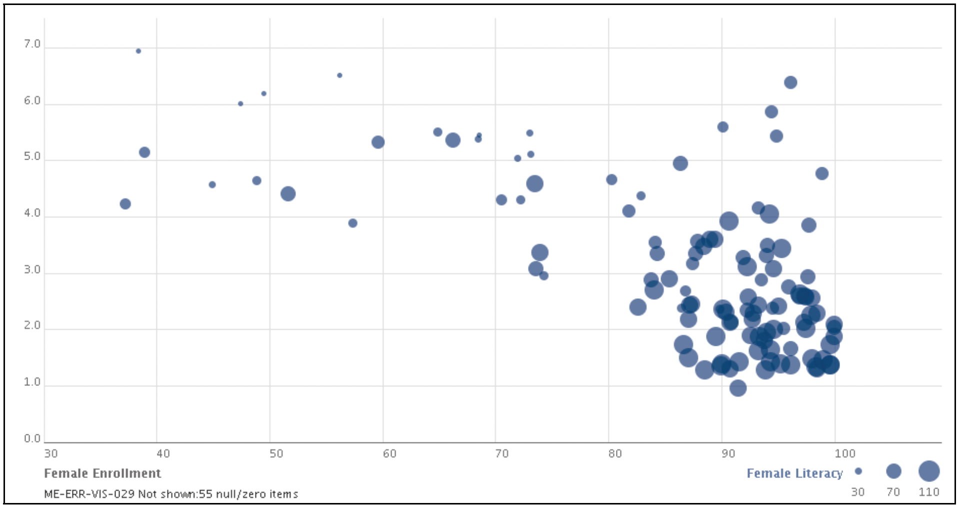


图3-8　Many Eyes的散点图

Many Eyes的优势在于其中的可视化数据图都是可交互的，而且有一些定制选项。比如说在散点图中就可以用第3种指标来测量各个数据节点，而且鼠标悬停到感兴趣的数据节点上时还能查看具体的数值。

除了传统图表和一些基础的地图工具之外，Many Eyes还提供了多种更为先进和试验化的可视化方法。单词树（word tree）可以帮助我们研究一整段文本，例如书或文章中的某个段落。首先选择一个单词或短语，然后就能通过与它相连的单词了解到它在整段文本中的使用状态。比如说，图3-9就显示了在美国宪法中搜索“right”一词得到的结果。

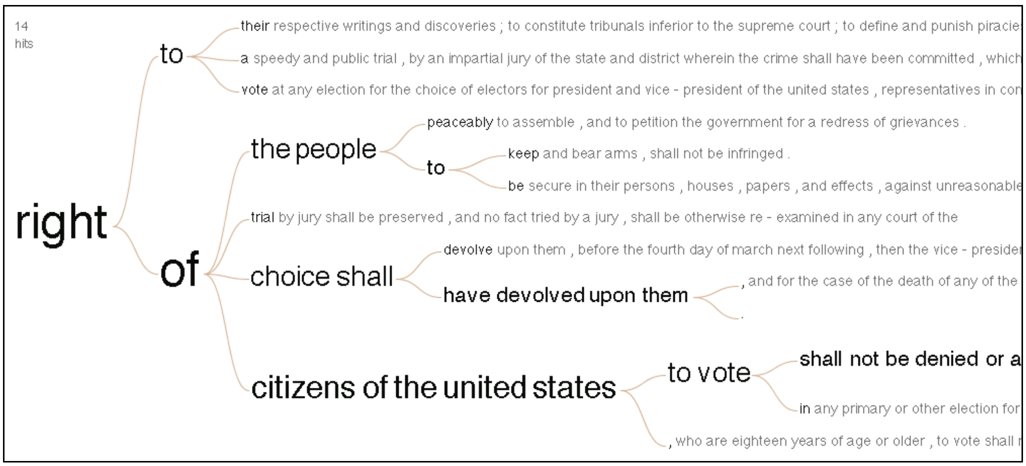


图3-9　Many Eyes的单词树显示了美国宪法的部分内容

针对同样的数据，我们还可以轻松地在各个工具之间转换。图3-10是一种特殊风格的单词云，被称为“Wordle”，同样显示了美国宪法的内容。单词使用得越频繁，字号就越大。



图3-10　美国宪法的Wordle

如你所见，Many Eyes有许多选项便于我们处理数据，而且它也是目前数据探索中用途最为广泛的免费工具（在我看来也是最好的）。不过有一两点仍然需要注意。第一是其中的大部分工具都是Java小应用程序，所以如果没有安装Java，可能就无法充分利用它。（对大多数人来说这其实不算什么问题，但我确实认识一些人，不管是什么原因，他们对自己电脑上安装了什么非常在意。）

另一点对多数人来说则可能更为敏感一些：上传到网站的数据是存储在公共空间里的。所以最好不要用Many Eyes来挖掘你的公司客户信息，或者销售数据。

▶访问http://many-eyes.com，试着上传你自己的数据并可视化它。

3

4. Tableau Software

Tableau Software是一款只面向Windows的软件，相对比较新，但在过去几年中开始受到越来越多的欢迎。它的设计初衷主要是用于视觉化的数据研究和分析。很明显该软件在美学和设计上花了不少心思，这也是人们喜欢它的原因。

Tableau Software提供了许多可交互的可视化工具，在数据管理方面也表现不错。数据可以从Excel、文本文件和数据库服务器中导入，生成标准的时间序列图表、柱形图、饼图、基本地图等多种图形。你可以挂接动态数据源，将各种图形混合搭配形成定制视图，或者通过仪表盘视图随时关注数据的状态。

最近，Tableau发布了Tableau Public，这是免费版本，包含桌面版的一部分功能。你可以把数据上传到Tableau的服务器，创建可交互的图形，然后轻而易举地发布到你的网站或博客上。不过和Many Eyes一样，上传到服务器的所有数据都是公开的，所以需要小心。

如果想用Tableau，但又想确保数据隐私，就需要购买桌面版本。本书写作时，桌面版本的售价是个人版999美元，专业版1999美元。

▶访问http://tableausoftware.com了解Tableau软件。网站提供了带完整功能的免费试用。

5. your.flowingdata

对个人数据收集的兴趣让我开发了自己的应用，your.flowingdata（YFD）。这是一款在线应用，用户可以从Twitter收集数据，通过一系列可交互的可视化工具探索其中的模式和联系。有人通过它来追踪自己的饮食习惯或者起居时间。还有人用它来记录自己婴儿的每日状况，稍微改动一下数据就成了一本宝贝剪贴簿。

YFD本来主要针对的是个人数据方面，但有些人发现这个应用对于通用类型的数据收集也很有帮助，例如网络活动监控或者火车时刻表。

▶访问http://your.flowingdata.com，尝试通过Twitter来收集个人数据。

3.1.2　取舍

尽管这些工具都非常容易使用，但也存在一些缺憾。只需鼠标操作无疑十分便捷，但随之而来的却是丧失一定的灵活性。你可以改变颜色、字体和标题，但仅限于软件所提供的那些元素。如果界面上没有你想要的图表按钮，你就只有唉声叹气的份。

另一方面，有些软件可能确实提供了充分的功能，但你可能需要学习大量的按钮。比如说，我曾花了一个周末去突击学习某款软件（前文并未列出），而且很明显，如果投入更多时间，我就能通过它完成很多事情。但是这款软件的整个操作流程完全匪夷所思，让人根本没有心思学下去。而且要想针对不同的数据集重复一遍流程也很困难，因为我必须记住之前点击过的所有按钮。而相比之下，通过代码来处理数据就会更加容易，因为针对不同的数据集只需稍微改动一下代码就可以解决。

不要误解我的意思。我并不是说这些开箱即用的软件就应该完全弃之不用。它们能帮你快速且方便地研究数据。但当你开始接触更多的数据集时，可能这些软件就不能满足所需了。到这个时候，你可以借助于编程手段。

3.2　编程工具

不用太紧张。掌握一点点编程技巧，你就能利用数据做更多的事情，远远超过那些开箱即用的软件。编程技巧能赋予你更加灵活的能力，而且各种类型的数据都能适应。

大多数设计新颖、令人惊艳的数据图都是通过代码或绘图软件实现的，很有可能两者兼有。有关绘图软件我们稍后也会谈到。

对于新手来说代码可能颇为神秘——我也是这样过来的。但我们可以把它当做一门新语言来看待，因为它确实如此。每一行代码都告诉计算机去做某件事情。计算机不懂我们和朋友之间所用的语言，所以你必须用它自己的语言或语法才能和它交流。

与任何语言一样，你不可能立刻就开始进行对话。要从基础开始，然后逐步建立自己的学习方式。很可能在你意识到之前，你就已经开始写代码了。关于编程最酷的事情在于，一旦你掌握了一门语言，学习其他语言就会更加容易，因为它们的逻辑思路是共通的。

3.2.1　可选项

看来你已经决定要卷起袖子写代码了——好样的。在这里你有很多选择。针对不同的任务，各种语言之间互有优劣。有些语言善于处理大量数据，其他一些则可能偏向于更好的视觉效果，或者提供交互功能。选择何种语言取决于你对数据图的目标，或者看你对哪种语言更习惯。

有些人坚持只学一门编程语言，并把它学得很透。这没问题，而且如果你是编程新手，我强烈推荐这一策略。尽量熟悉代码的基础规范和重要概念。

我们应该选择最能满足自己需要的编程语言。不过，掌握新语言、了解新玩法也是很有乐趣的。因此在作决定之前，不妨先积累一些编程经验。

1. Python

前一章我们已经讨论了如何利用Python来处理数据。Python善于处理大批量的数据，不会造成宕机。这使得该语言能够胜任繁重的计算和分析工作。

Python干净易读的语法也很受程序员们欢迎，还可以利用很多模块来创建数据图形，例如图3-11中的这种。

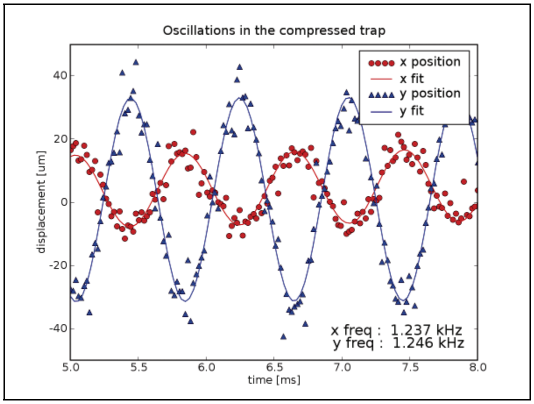


图3-11　利用Python生成的图表

从美学方面来看，这个图表还不够好。直接拿Python输出的图片用于印刷可能会比较勉强，尤其是在边缘处给人感觉比较粗糙。但不管怎样，这是数据探索阶段一个很不错的开始。或者你也可以先输出图片，然后再利用其他的图形编辑软件来润色或添加信息。

有用的Python资源

- Python官方网站http://python.org。
- NumPy和SciPy[①]http://numpy.scipy.org/——科学计算模块。

2. PHP

PHP是我刚开始网页编程时学到的第一门语言。有些人说它很松散，确实如此，但也可以让它很有条理。大部分Web服务器都预安装了PHP的开源软件，因此要想着手写PHP是非常容易的。

绝大多数预安装中都会包含一个叫做GD的图形函数库。这个库非常灵活，能让你从无到有地创建图形，或者修改已有图形。此外还有很多PHP图形函数库能帮助我们创建各类基本的图表。最受欢迎的是Sparkline（微线表）库，它能让你在文本中嵌入小字号的微型图表，或者在数字表格中添加视觉元素，如图3-12所示。

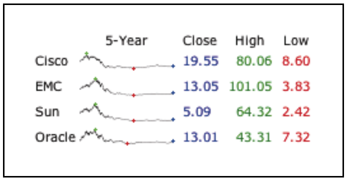


图3-12　利用PHP图形函数库生成的微线表

① NumPy是Python的一个数据处理的函数库，里面主要是一些矩阵的运算等。SciPy是Python语言中用于科学研究的函数库，它是在Numpy基础上开发的。

一般PHP的出现都会伴随着MySQL等数据库，而不是一堆CSV文件。这使它能物尽其用，处理大型的数据集。

有用的PHP资源

- PHP官方网站（http://php.net）。
- Sparkline PHP图形函数库（http://sparkline.org）。

3. Processing

Processing是一门适合于设计师及数据艺术家的开源语言。最早的Processing还只能算作是小品级别，可以让用户快速生成图形，但随后获得了长足的发展，完成了很多高质量的项目。比如说，第1章中提到的*We Feel Fine*就是用Processing中创建的。

Processing很棒的一点是能很快上手：轻量级的编程环境，只需几行代码就能创建出带有动画和交互功能的图形。这款工具确实很基础，但由于它偏重于视觉思维的创造性，你很容易就能知道如何创造出更高级的作品。

虽然在一开始主要是设计师和艺术家使用Processing，但如今它的受众群体已经越来越多样化了。你可以借助各种函数库来提升它的威力。

它的缺点之一是要使用到Java小应用程序，在某些计算机上载入时可能会很慢，而且并不是每个人都安装了Java（尽管多数人都安装了）。不过这也有解决办法，Processing在不久前发布了它的JavaScript版本。

无论如何，它对于新手来说是个很好的起点。即使是毫无编程经验的用户也能够做出有价值的东西。

有用的Processing资源

- Processing的官方网站（http://processing.org）。

4. Flash和ActionScript

网上大多数可交互的动画数据图都是通过Flash和ActionScript开发的，尤其是在《纽约时报》这样的主流新闻网站上。我们可以直接用Flash来设计图形，这也是一款所见即所得的软件，但有了ActionScript的帮助，就能更好地控制交互行为。许多应用都是完全用ActionScript写的，无需用到Flash环境，不过这些代码还是作为Flash应用来进行编译。

说明　虽然有很多免费、开源的ActionScript函数库，但是Flash软件和Flash编译器的价格比较昂贵，在你选择软件时需要考虑这一点。

比如说，表现沃尔玛企业成长的可交互动画演示地图（见图3-13）就是用ActionScript写成的。其中调用了Modest Maps库。这是一个用于区块拼接地图（tile-based map）的显示和交互的函数

库，以BSD许可协议[①]发布，这表示它是免费的，大家可以随心所欲地使用。

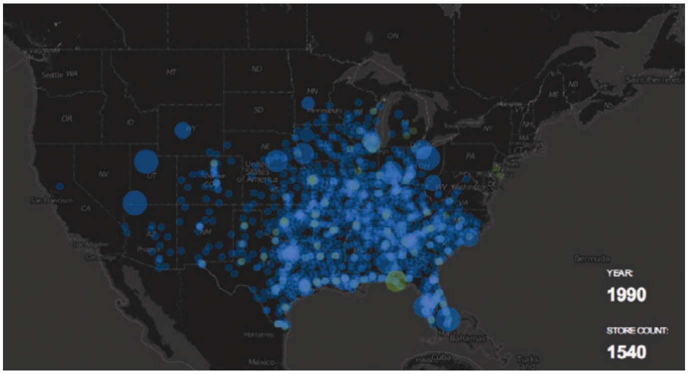


图3-13　动画演示沃尔玛企业成长的地图，以ActionScript写成（另见彩插图3-13）

图3-14中的可交互堆叠面积图也是由ActionScript写成的。它能让你搜索数年来各种消费开支的大小变化，例如住房（Housing）和交通（Transportation）。这一艰巨的任务是由加州大学伯克利分校可视化实验室开发的Flare Action Script库完成的。

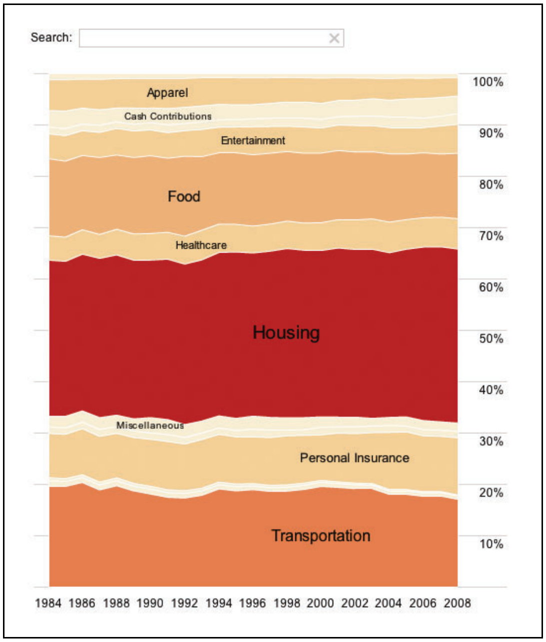


图3-14　显示消费开支分类的可交互堆叠面积图，以ActionScript写成

① BSD许可协议（Berkeley Software Distribution license），是自由软件（开源软件的一个子集）中使用最广泛的许可协议之一。

如果你对网上的可交互图形感兴趣，那么Flash和ActionScript是非常好的选择。Flash应用的下载相对较快，而且绝大多数人的计算机上都安装了Flash播放器。

不过ActionScript并不好学。虽说语法并不复杂，但是安装和代码的组织却可能会难倒初学者。你不可能像Processing那样只用几行代码就运行成功。在后面的章节中我会带领大家实践其中的基本步骤，另外由于Flash的广泛使用，在网上还能找到很多极有帮助的教程。

与此同时，Web浏览器一直在飞速发展，其运行速度和效率也有了显著的提高。这让我们有了很多其他的选择。

有用的Flash和ActionScript资源

- Adobe支持（http://www.adobe.com/products/flash/whatisflash/）——Flash和ActionScript（以及其他Adobe产品）的官方文档。
- Flare可视化工具包（http://flare.prefuse.org）。
- Modest Maps函数库（http://modestmaps.com）。

5. HTML、JavaScript和CSS

Web浏览器的运行速度越来越快，功能也越来越完善。不少人使用浏览器的时间要超过计算机上的其他任何程序。可视化在近期也有了相应的转变，开始借助HTML、JavaScript和CSS代码直接在浏览器中运行。在过去，可交互的数据图一般都是通过Flash和ActionScript来实现，而静态数据图则需要存储为图片格式。现在的情况也大抵如此，但不再只有这一种选择。

一些功能强健的工具包和函数库可以帮助我们快速创建可交互或静态的可视化图形。它们还提供了大量的选项，以便你针对数据需要进行定制。

比如说，斯坦福大学可视化团队开发的Protovis就是一款免费开源的可视化函数库，它能帮助你创建网页原生的可视化作品。Protovis提供了一系列开箱即用的可视化工具，但你在创建几何图形时不会受到任何限制。图3-15显示的是一幅可交互的堆叠面积图。

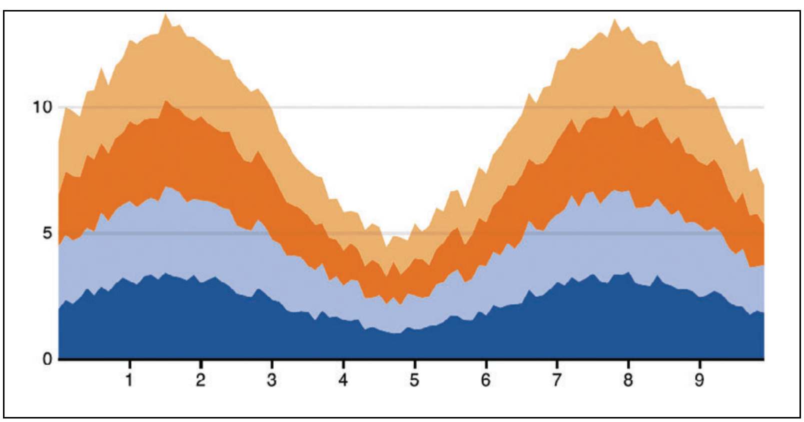


图3-15 由Protovis生成的堆叠面积图

Protovis中内嵌了这一图表类型。我们还可以生成形式更加新颖的流线图（streamgraph），如图3-16所示。

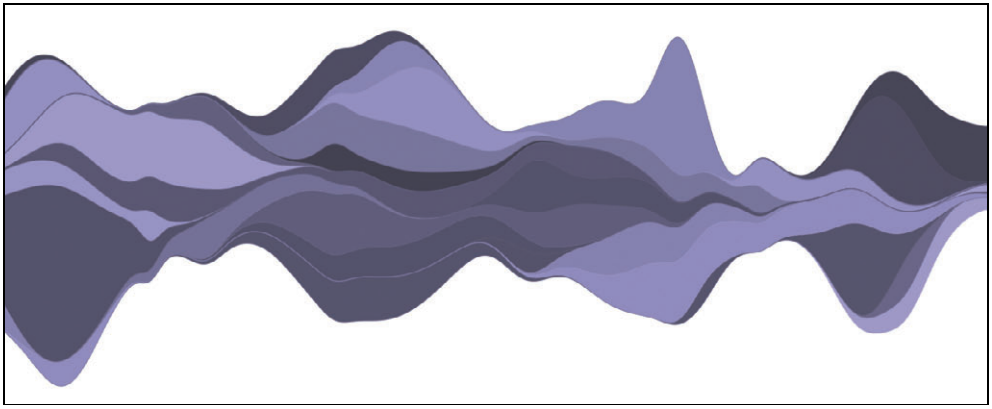


图3-16　由Protovis自定义生成的流线图

我们还可以利用多个库来扩充功能。Flash也能做到这一点，但JavaScript在代码方面的工作量会小很多。有了jQuery和MooTools等库的协助，JavaScript也更加容易阅读和使用。这些库并非专为可视化而开发的，但仍然能带来很大帮助。它们提供了大量基础功能，只需几行代码就能实现。如果没有这些库，我们要写的代码就会更多，而且稍不注意就可能漏洞百出。

这些库还提供了一些插件，帮助我们制作较为基础的图形。例如，你可以利用jQuery的Sparklines（微线表）插件来生成微线表（见图3-17）。

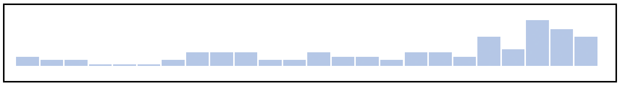


图3-17　通过jQuery的Sparklines插件生成的微线表

用PHP也可以做到这一点，但这种方法具有更多优势。首先，数据图是在用户的浏览器中生成的，而非服务器端。这能缓解服务器的压力，否则在流量较大的情况下你的网站就可能会出问题。

另一个优势在于你无需在自己的服务器上安装PHP图形库。很多服务器上都预安装了这些图形库，但也有一些没有。如果你对PHP图形库不熟悉，安装过程可能会非常麻烦。

也许有人不想用插件。没关系，利用标准的网页编程也可以定制可视化的效果。图3-18就是一副可交互的日历，同时也是用户使用your.flowingdata工具的热度图（heatmap）。

不过还是有几点需要注意。由于相关的软件和技术还比较新，在不同浏览器中你的设计可能在显示上会有所差别。在Internet Explorer 6这类老旧的浏览器中，有些工具可能无法正常运行。不过这种情况正在逐渐好转，因为绝大多数人都已经转向了Firefox或Google Chrome等更现代的浏览器。归根结底，这还是取决于你的受众群体。在FlowingData网站的访问者中，只有不到5%的人还在用低版本的Internet Explorer，因此浏览器兼容性并不是很严重的问题。

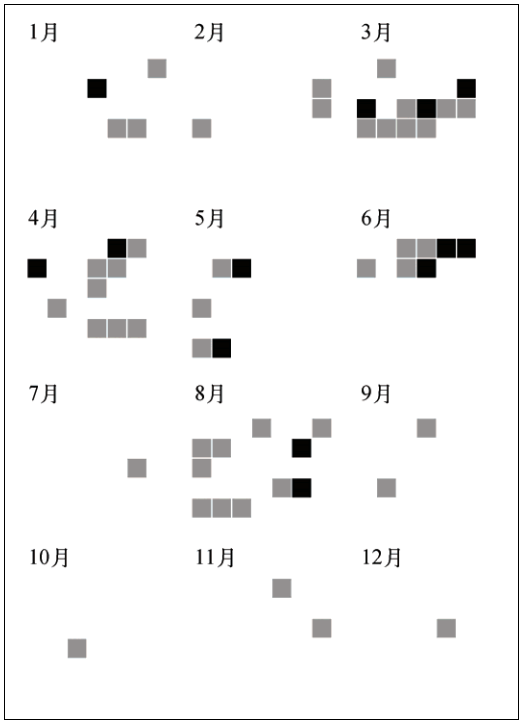


图3-18　可交互日历，同时也是用户使用your.flowingdata的热度图

同样，由于该技术的普及率尚不够高，为JavaScript可视化提供支持的函数库不像在Flash和ActionScript中那么多。这也是为什么许多主流新闻机构仍然大量使用Flash的原因。不过随着技术的发展这一局面终会发生改变。

有用的HTML、JavaScript和CSS资源

- jQuery（http://jquery.com/）——一个JavaScript库，能让该语言的编程更加高效，而且让最终代码更加易读。
- jQuery Sparklines（http://omnipotent.net/jquery.sparkline/）——通过JavaScript 生成静态及动画的微线图。
- Protovis（http://vis.stanford.edu/protovis/）——专用于可视化的JavaScript库，提供了实例以便学习。
- JavaScript InfoVis工具包（http://datafl.ws/15f）—— 另一个可视化库，但不如Protovis成熟。
- Google Charts API（http://code.google.com/apis/chart/）——动态创建传统形式的图表，只需修改URL即可。

6. R

如果你浏览过FlowingData，可能就会知道我最喜欢的数据图形软件就是R。它是一款免费且开源的统计学计算软件，图形功能也很强大。它也是绝大多数统计学家最中意的分析软件之一。此外还有一些功能近似的付费软件，例如S-plus和SAS，不过它们很难比得上R的完全免费以及活跃的开发社区氛围。

相较于之前提到的所有软件，R的优势之一在于它是专为数据分析而设计的。HTML的用途主要是创建网页，而Flash则用于其他方面，例如视频以及动态广告。但R却是由统计学家开发并维护，主要的面向对象也是统计学家。至于这是好是坏，取决于你从哪种角度来看待它。

支持R的工具包也有很多，你只需把数据载入到R里面，写一两行代码就可以创建出数据图形。比如说，你可以利用Portfolio工具包快速创建出板块层级图（treemap），如图3-19所示。

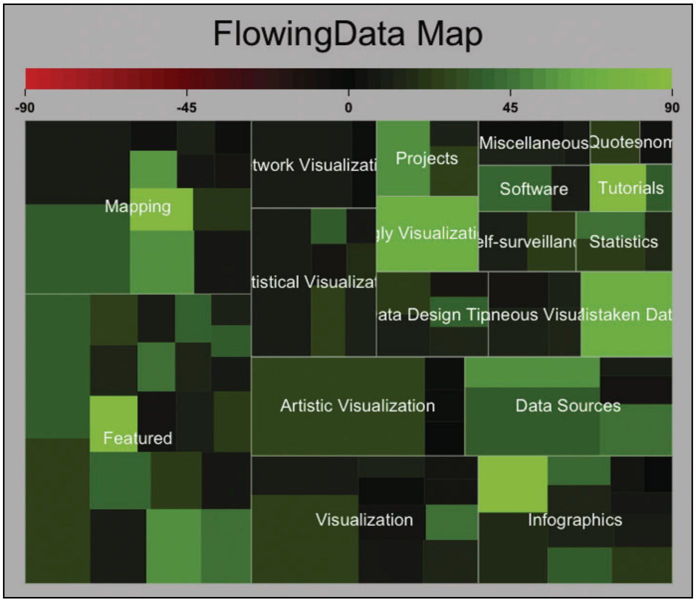


图3-19　在R中利用Portfolio工具包生成的板块层级图（另见彩插图3-19）

创建热度图也同样简单，如图3-20所示。

当然，还可以创建更多传统的统计图表，例如散点图和时间序列图，我们将会在第4章中对此进行深入讨论。

坦白说，R的网站看起来极为落后（见图3-21），而且软件本身在引导新用户方面做得不够好。不过你要记住，R毕竟是一门编程语言，学习任何语言都必须经历开头的艰难。我也读到过一些关于R的负面评论，但通常都是那些习惯于按钮或者鼠标操作的人在发牢骚。所以如果你打算使用R，不要对用户界面的期待过高，否则你一定会觉得它很不友好。

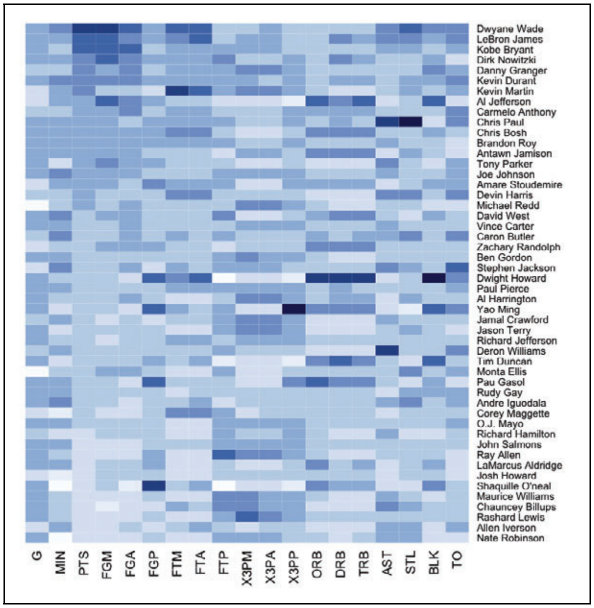


图3-20　R生成的热度图

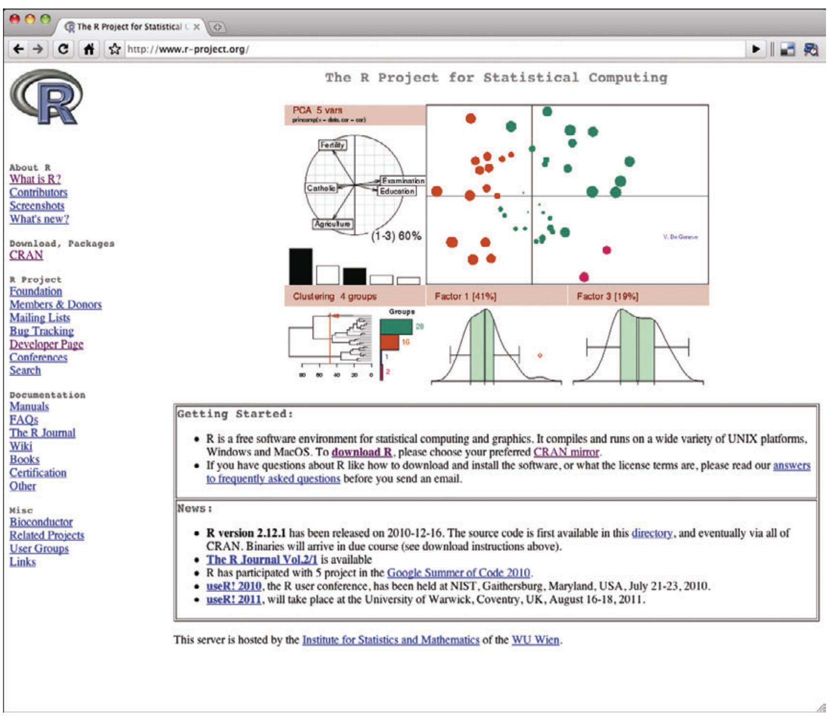


图3-21　R的网站首页http://www.r-project.org

但如果克服了这一点，我们就能用R做很多事情。它能够生成达到印刷品质的图片（至少也是雏形），而且极其灵活。如果愿意，你可以写出自己的函数或者程序包，按自己想要的方式来创建图形，或者你也可以借用R函数库里其他人开发的成品。

R提供了基础的绘图功能，使用它，你基本上可以随心所欲地绘制想要的图形，例如线条、形状以及框架坐标轴。所以和其他编程工具一样，能限制你的只有想象力而已。事实上，任何一种图表类型都能通过R或者R的工具包实现。

提示 在搜索引擎中搜索有关R的内容时，它的本名可能会带来很多不相干的结果。可以尝试以“r-project”作为关键词，这样的搜索结果会更加精确。

既然R这么强大，为什么还要学习其他工具呢？为什么不干脆用R来做所有事情？原因有以下几方面， R是在你的桌面上运行的，所以它不太适合于动态网页。存储为图片然后发布到网页上并不是问题，但这一过程不会自动完成。你也可以通过网页来动态生成图片，但截至目前，R的这一功能还不是特别强大，无法比拟JavaScript等网页原生工具。

在创建可交互图形或动画方面，R也不是特别擅长。同样，尽管也可以用R来实现，但还有其他更方便的途径，比如Flash或者Processing。

最后，大家也许已经注意到，图3-19和图3-20中的图形还欠缺一些雕琢。你恐怕不希望在报纸上看到这种水平的图形。当然，你也可以尝试依靠不同的选项或者添加代码来强化R的设计感觉，但我的策略一般是在R中创建基本图形，然后再利用Adobe Illustrator等绘图软件对其进行编辑加工（后文即将讨论）。如果是用于数据分析，那么R的原始输出就可以了，但如果是用于演示或印刷，则最好还是从视觉和美感上稍加调整。

有用的R资源

- 适用于统计运算的R项目（http://www.r-project.org）。

3.2.2 取舍

学习编程就是在学一门新的语言，一门有关位和逻辑的计算机用语言。例如，当你使用Excel或Tableau时，本质上是在和程序的翻译器打交道。按钮和菜单是你熟悉的语言，而当你点击它们时，软件会把你的行为翻译为指令，并发送给计算机。然后计算机才会服从你的命令，例如生成图片或者处理数据。

所以，时间绝对是主要的障碍。学习一门新的语言需要时间。对很多人来说，这一障碍太高了，对我来说也是如此。你需要立刻完成工作，因为你面前有一大堆数据，而且还有人等着要结果。如果你只会与数据打这么一次交道、以后不会再接触，那么选择开箱即用的可视化工具可能会更合适。

不过，如果你打算深入研究你的数据，而且日后可能（或者希望日后）还会接触大量与数据相关的项目，那么现在花些时间学习编程最终会节省其他项目的时间，并且作品也会给人留下更加深刻的印象。你的编程技巧会在每一次项目中获得提高，你会发现编程越来越容易。和任何一门外语一样，我们都会从基础学起，然后再逐步扩大范围，不可能从一开始就用它来著书立说。

让我们换一种思路来看这个问题。假设你被扔到一个陌生的国度，你不会说当地的语言，但有一名随身翻译。如果要和当地人交流，你会先说出来，然后让翻译去传递你要表达的信息。如果翻译不理解你的意思，或者不懂你用的某一个词，又该怎么办？他可能会干脆忽略这个词，或者如果他足够聪明，说不定去查查字典。

对于开箱即用的可视化工具来说，软件本身就是翻译。如果它不知道该怎样做一件事情，你就只能卡在那里，或者不得不寻找替代的解决办法。和现实中的翻译不同，软件通常不会立刻学会“新词”（在这里指新的图表类型或者数据处理方式）。新的功能需通过软件更新获得，也就是说你只能干等着。那么，如果你自己学会了这门语言会怎样呢？

再一次声明，我并不是说要避免使用开箱即用的工具。我一直都在使用它们。它们让很多沉闷乏味的任务变得轻松容易，这很棒。但我们不能被软件限制住，不是吗？

在后面几章大家就会看到，编程能够帮助我们做很多事情，而且付出的努力要比徒手完成少得多。不过，还是有很多事情必须依靠双手来完成，尤其是当你打算通过数据讲故事的时候。这也就是下文的内容：可视化的另一面。

3.3　绘图软件

本节内容对于图形设计师来说应该是非常熟悉的了。但如果你是一名分析师，或者身处技术岗位，你可能就会发现自己进入了一个陌生的领域。代码和开箱即用的可视化工具确实能帮助我们实现很多想法，但得到的数据图却总带有自动生成的感觉：要么标记的位置不合适，要么就是说明文字杂乱不堪。如果只是用于分析，这通常也没有问题，毕竟你知道图中元素的意义。

但是，如果你制作的图形要用于演讲、报告或者印刷出版，那么经过雕琢的数据图无疑更加合适，因为这样人们才能更清楚地理解你要讲的故事。

比如说，图3-19是R直接输出的图形。它显示了FlowingData网站最受欢迎的100篇文章的浏览次数和评论数量。所有文章都以矩形显示，并进行了分类，例如Mapping（地图）类和信息图（Infographics）类等。绿色越明亮（对应于灰度越浅）表示该文章获得的评论越多，矩形面积越大表示该文章被浏览的次数越多。从原始的图形中你看不出这些，只有我自己知道各个元素的意义，因为R里的代码是我写的。

图3-22是它的修订版本。标签都进行了调整以便阅读，顶部添加的说明文字告诉读者自己正在看什么。颜色说明中的红色部分也被去掉了，因为不可能有负评论数的文章。此外我还把背景从灰色改成了白色，这样看起来更加顺眼。

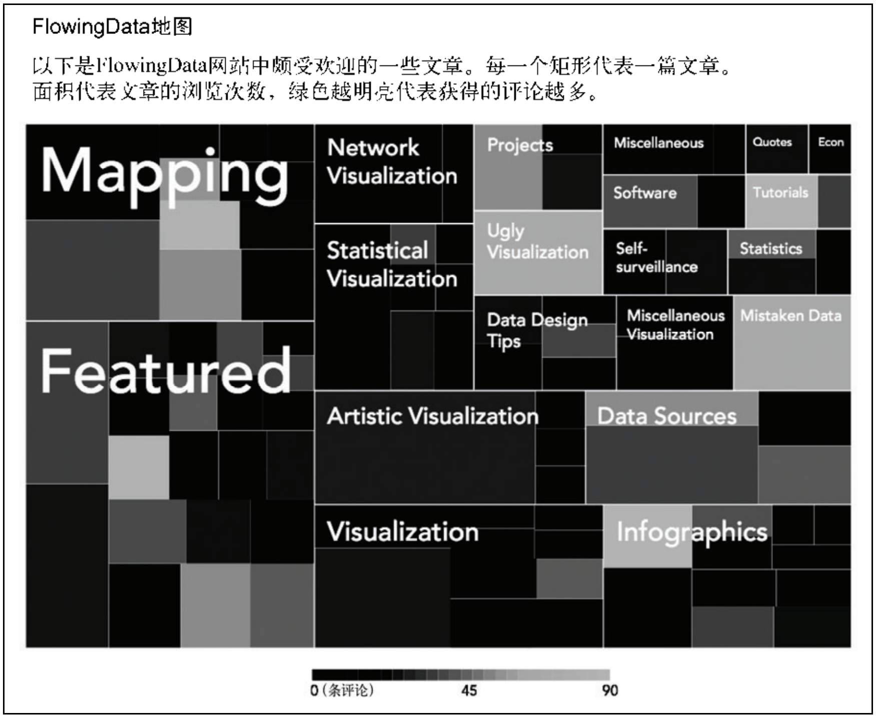


图3-22　在R中创建、在Adobe Illustrator中编辑后的板块层级图

我也可以靠写代码来实现以上改动，但在Adobe Illustrator里面直接用鼠标操作要容易得多。你可以完全用绘图软件来创建图表，也可以先用R或其他工具生成图表，然后再导入到设计软件中加工。前者的可视化类型会受到一定限制，因为这些软件的主要用途并不是可视化。要想创建比柱形图更复杂的图表，那么最好还是选择后者，否则你需要手工做很多事情，而且容易产生错误。

绘图软件的优势在于你可以更好地控制单个元素，而且只需要鼠标操作。你可以随心所欲地改变柱形图的颜色、调整坐标轴的宽度或者添加注释。

3.3.1　可选项

市面上有很多绘图软件可供选择，但人们常用的就有那么几款——而且有一款是几乎所有人都在用的。价格应该是决定性因素，这些软件的价格从免费（且开源）到数百美元不等。

1. Adobe Illustrator

任何看上去像是量身定做、或者出现在主流新闻出版物中的静态数据图，都极有可能在一定程度上经过了Adobe Illustrator的处理。Adobe Illustrator是设计业界的标准。《纽约时报》中送印的每一幅图表都是在Illustrator中创建或编辑的。

Illustrator被广泛用于印刷是因为它处理的是矢量图形，而非像素。这意味着你可以将图片无

限放大，而不会损失显示质量。相对地，如果你放大的是低分辨率的照片（照片都是由固定数量的像素组成），那么就会发现图片出现严重的失真。

这款软件最初是为了方便字体设计而开发的，后来在图形设计师当中受到欢迎，用于logo设计或艺术图形绘制。如今这仍然是Illustrator的主要用武之地。

不过，Illustrator也提供了图表工具（Graph tool），可以生成一些基础的图表。我们可以把数据粘贴到软件的小型电子表格中，然后自动生成柱形图、饼图、时间序列图等基础图表。不过需要注意的是，数据量不能超出Illustrator的能力范围。

从数据图角度来说，Illustrator的优势在于其灵活性，以及丰富的功能所带来的方便性。丰富的功能同时也带来了更复杂的界面，所以新手会感觉难以上手，但用不了多久就会适应。第4章将会涉及这些内容。而正是因为Illustrator功能的灵活与丰富，数据设计师们才能创造出各种简洁明了的优秀图表。

Illustrator支持Windows和Mac系统。它的缺点是，与用代码完成所有事情相比较而言，价格比较昂贵，后者是免费的，只要你在电脑上安装好合适的工具就行。不过如果和一些开箱即用的工具相比，说实话Illustrator也贵不到哪里去。

写作本书时，Adobe官网上最新版Illustrator的标价是599美元。不过大家应该能在别处找到打大幅折扣的机会（或者购买稍旧的版本）。Adobe对学生和学术界人士都有较大的优惠，大家可以根据实际情况考虑。（它是我买过的最贵的软件，但我几乎每天都会用到它。）

有用的Adobe Illustrator资源

- Adobe Illustrator官方产品页面（http://www.adobe.com/products/illustrator/）。
- VectorTuts（http://vectortuts.com）——提供大量Illustrator的简明使用教程。

2. Inkscape

Inkscape是Adobe Illustrator的免费且开源的替代产品。如果你想省下这一大笔开支，那么Inkscape是你最好的选择。我之所以使用Illustrator，是因为在我刚开始学习数据图时，周围人全都用它，自然我也受到了影响。不过我经常听到有关Inkscape的好评，而且因为它是免费的，所以试一下没有坏处。只是不要期望有很多教程资源。

提示　本书中主要通过Adobe Illustrator来加工数据图，不过要想在Inkscape里学会做同样的事情应该也不会太难。其中许多工具和功能的命名都是相似的。

有用的Inkscape资源

- Inkscape官网（http://inkscape.org）。
- Inkscape教程（http://inkscapetutorials.wordpress.com/）。

3. 其他软件

当然，Illustrator和Inkscape并不是我们的唯一选择，只不过有很多人用它们罢了。也许你习惯于其他软件，比如Corel Draw。Corel Draw是一款只支持Windows系统的软件，价格和Illustrator大致相当，有些地方可能会卖得稍微便宜一点。

此外还有一些程序会提供规模较小的工具集，例如Aviary and Lineform出品的Raven。Illustrator和Inkscape属于图形设计师的综合性工具，所以它们具备的功能较多。但如果你只是想对已有图形进行少量编辑，不妨选择更加简单（也更加便宜）的软件。

3.3.2 取舍

绘图软件的目的就是方便人们绘图。它们并非专用于数据图，而是针对整个图形设计领域，很多人会发现在Illustrator和Inkscape里有不少功能都用不上。这类软件在处理大型数据方面也比不上编程或专业可视化工具。因此，你也不能利用它们探索和分析数据。

不过如果你希望制作出用于印刷的数据图，这类软件是必不可少的。它们不仅能提升美感，而且还能提高可读性和清晰度，而这些都是自动生成的图片难以企及的。

3.4 地图绘制工具

用于绘制地理数据的工具与上文的可视化工具之间存在一些交集。不过随着近年来地理数据的显著增多，我们绘制此类数据图的手段也层出不穷。如今移动定位服务持续升温，随之也将出现更多带有经纬度坐标的数据。地图也是一种非常直观的数据可视化方式，值得我们关注。

在互联网发展的早期，绘制电子地图并不容易，效果也谈不上讲究。还记得当年在访问MapQuest网站时，必须先找到东南西北，然后才能得到一小幅静态地图吗？Yahoo!也曾提供过类似的服务。

这一情况直到数年后Google实现了顺滑地图（slippy map）技术后才有所改观（见图3-23）。顺滑地图技术存在了好一段时间，但直到人们的网速快到足够应付地图的连续更新，它才体现出真正的价值。我们今天使用的地图服务都用到了这一技术。除了能轻松地移动、缩放地图以便查找方向之外，地图也是浏览某些数据集的主要接口。

说明 顺滑地图技术是一种如今广泛使用的地图实现技术。大型的地图通常都会超出显示器屏幕的范围，因此会被分割为较小的图片，或者说区块（tile）。只有在显示器区域内的区块会显示，其他的都被隐藏起来。当用户拖动地图时，其他区块才会显示，给用户“正在拖动一整幅地图”的感觉。大家在查看高分辨率照片时可能也曾遇到过类似的处理方式。

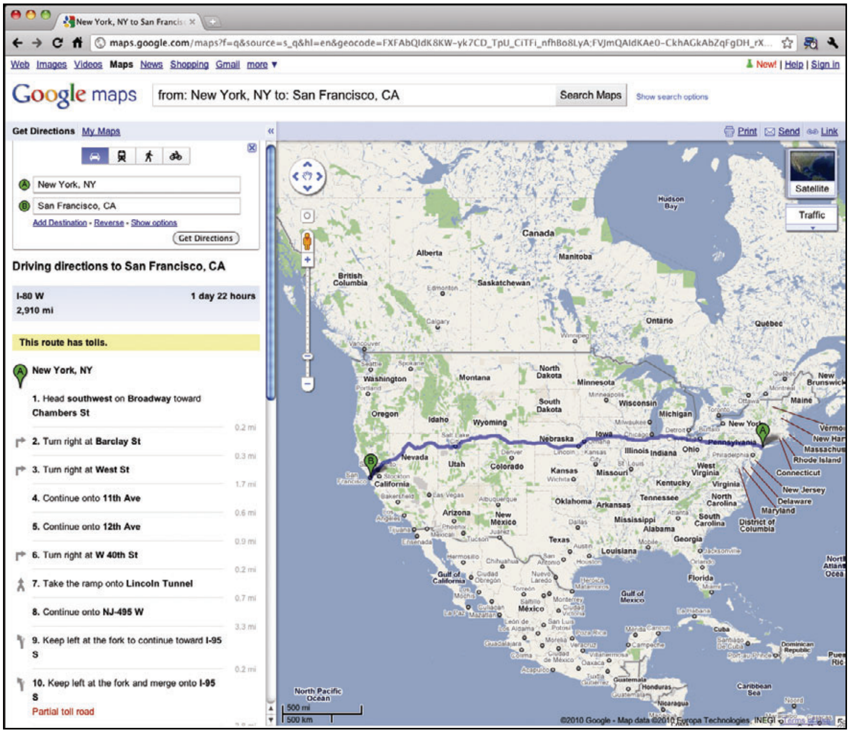


图3-23　利用Google Maps来查找方向

3.4.1　可选项

随着各类型的地理数据纷纷涌入到公众领域，绘制这些数据的工具也开始涌现。其中一些只需要编写一点点代码就能运行，另外一些则需要多花些工夫。此外，还有一些工具根本就不需要编程。

1. Google、Yahoo!和Microsoft地图

这是最容易的在线工具，不过需要一些编程技巧才能实现。你的编程能力越强，就能通过Google、Yahoo!和Microsoft提供的地图API来完成越多的事情。

这三者的基本功能都很相似，但如果你是新手，我会推荐Google，因为它似乎最为可靠。它提供了基于JavaScript和Flash的地图API，以及其他一些地理相关服务，例如地形编码和方向查找等。你可以从入门指南开始，然后再扩展到其他功能，例如放置标记（见图3-24）、绘制路径或者添加覆盖图等。面面俱到的教程和代码片段应该能让你很快上手。

Yahoo!也提供了基于JavaScript和Flash的地图API，以及一些地理相关服务，但就该公司的现状来看，我不确定它还能支撑多久。在本书写作时，Yahoo!的角色已经由积极的应用开发者转变为单纯的内容提供商。Microsoft同样提供了一个JavaScript的API（Bing旗下的产品）和一个Silverlight支持的API（Microsoft的Flash版本）。

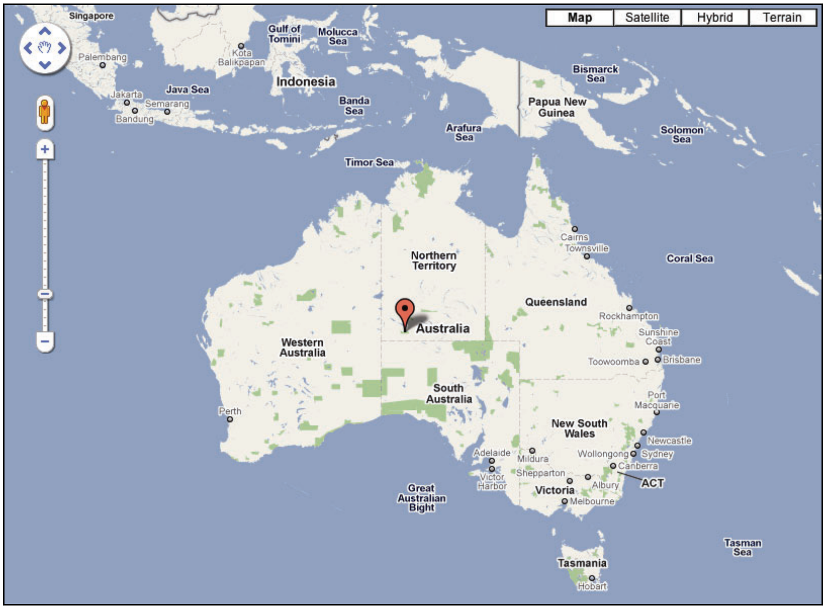


图3-24　在Google 地图上放置标记

有用的地图API资源

- ❑ Google地图API系列（http://code.google.com/apis/maps/）。
- ❑ Yahoo! 地图Web服务（http://code.google.com/apis/maps/index.html）。
- ❑ Bing地图API（http://www.microsoft.com/maps/developers/web.aspx）。

2. ArcGIS

从核心上来看，之前提到的在线地图服务提供的只是一些基本的功能。如果你想从中得到更高级的地理数据图，恐怕就只能靠自己来实现某些功能。而桌面地图软件ArcGIS则正好相反。它是一款大型软件，可以处理大批量的数据，还提供了平滑、合成等众多其他功能。你可以在用户界面中完成所有的事情，所以也不需要写代码。

任何配有地图绘制人员的图形设计部门基本上都在用ArcGIS，专业的地理制图师也在用ArcGIS。还有很多人喜欢它。如果你对制作详尽的地理数据图感兴趣，那么ArcGIS值得一试。

我只在部分项目中用过ArcGIS，因为只要情况允许，我一般都倾向于靠编程来实现，而且我不需要它提供的所有功能。功能丰富带来的不利的一面是按钮和菜单元素过多。你也可以使用它的在线和服务器解决方案，但相较于其他实现方式而言，这些都显得有些笨拙。

有用的ArcGIS资源

- ❑ ArcGIS的官方产品页面（http://www.esri.com/software/arcgis/）。

3. Modest Maps

我在图3-13的例子中曾提到过Modest Maps，它显示了沃尔玛的成长史。Modest Maps是一个Flash和ActionScript的区块拼接地图函数库，并且支持Python。它由一群了解在线地图的人维护，而且他们不仅是因为客户而工作，还因为这是他们的爱好。从这个角度你应该能大概了解到这个函数库的质量。

Modest Maps更像一个框架，而不仅只是一个地图API（类似Google提供的那种）。它只提供极少的必备条件，方便你创建在线地图，然后就退到一边让你随心所欲地实现。你可以使用来源不同的地图区块，也可以根据需要定制。比如说，图3-13是黑色与蓝色的主题，但你也可以轻松地改成红色和白色（对应于深色和白色），如图3-25所示。

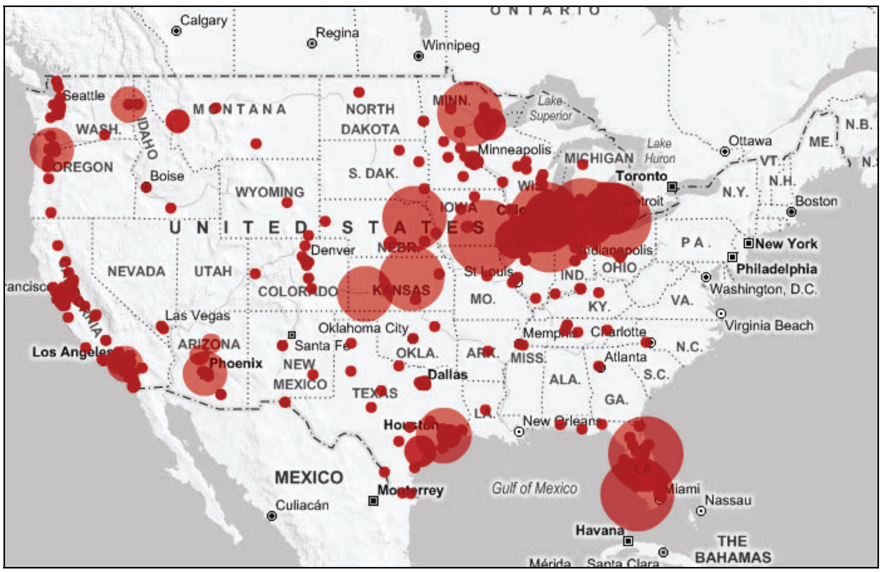


图3-25　利用Modest Maps创建的红白主题地图

Modest Maps是以BSD许可协议发布的，所以你可以免费用它来做任何事情。不过你还是需要对Flash和ActionScript有所了解，相关的基础知识将会在第8章介绍。

4. Polymaps

Polymaps有点像JavaScript版本的Modest Maps。它由同一群人中的一部分人开发及维护，提供同样的功能——而且还不止这些。Modest Maps只能进行基础的地图绘制，而Polymaps却有一些内置的功能，例如区域密度图（choropleth，见图3-26）和气泡图。

由于是JavaScript，它给人的感觉更加轻便（因为需要的代码更少），不过只能支持现代浏览器。Polymaps通过可缩放矢量图形（Scalable Vector Graphics，SVG）来显示数据，所以不能在老版本的Internet Explorer中运行，但大部分人都是与时俱进的。FlowingData网站只有5%的访问者还在用低版本的浏览器，而且我估计这一比例很快就会下降为零。

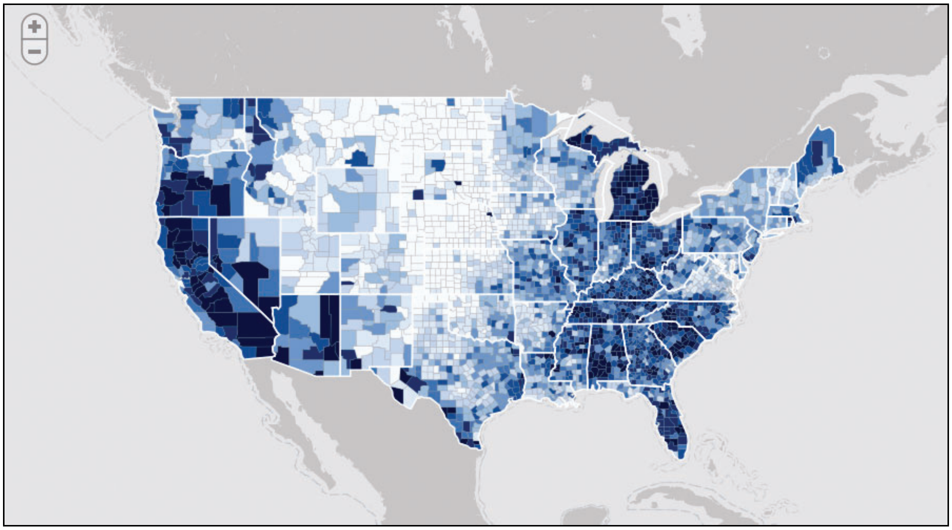


图3-26 显示失业率的区域密度图，通过Polymaps实现

我喜欢JavaScript地图函数库的另一个原因是所有的代码都在浏览器内原生运行。你无需任何编译或者Flash输出，因此执行起来更加容易，而且也便于日后更新。

有用的Polymaps资源

- ❑ Polymaps官方网站（http://polymaps.org/）。

5. R

R的基础版本中并未提供地图绘制功能，但有一些工具包能帮助我们。图3-27就是我在R中绘制的一张地图。注释是后期在Adobe Illustrator中添加的。

R中的地图在使用上存在限制，而且相关的文档也不够详尽。因此我只在数据足够简单，而且碰巧正在使用R时才会用它来创建地图。在其他的情况下，我会使用上述的工具。

有用的R地图资源

- ❑ 地区数据分析（http://cran.r-project.org/web/views/Spatial.html）——有关地区分析的综合性R工具包列表。
- ❑ 地理统计制图实用指南（http://spatial-analyst.net/book/download）——关于如何使用R及其他工具分析空间数据的可免费下载的电子书。

6. 在线工具

还有一些在线工具也能帮你可视化地理数据。在多数情况下，它们会调用人们最常使用的地图类型，然后把不重要的内容剥去——有点类似简化版的ArcGIS。免费的工具有Many Eyes和

GeoCommons，前者我们曾经讨论过，它只有一些基础功能，而且只能以国家或美国各州为单位。GeoCommons的功能更多，而且交互方式更加多样化，此外还能处理常见的地理文件格式，例如shapefile和KML。

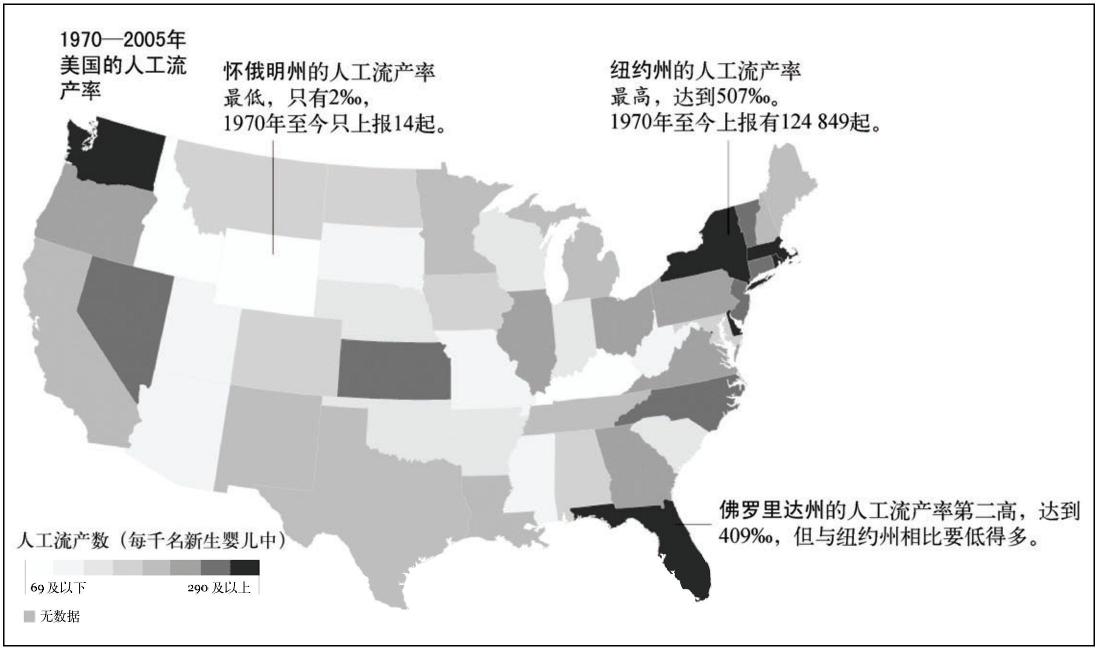


图3-27　R中创建的美国地图

此外还有一些付费工具，其中Indiemapper和SpatialKey是最有用的。SpatialKey更偏向于业务分析和决策制定，Indiemapper则更适合制图员和设计师。图3-28就是我用Indiemapper生成的例子，只花了几分钟时间。

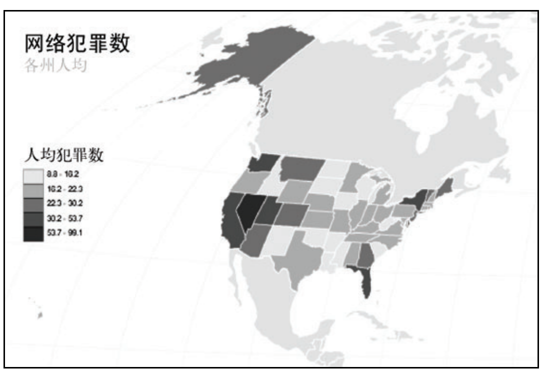


图3-28　Indiemapper中创建的区域密度图

3.4.2 取舍

各种形式的地图工具能满足我们不同的需求。只学会一款软件，然后就能设计出各种地图无疑是最理想的，但很遗憾这种方法行不通。

比如说，ArcGIS的功能很丰富，但如果你只需创建简单的地图，就没必要花时间去学习它复杂的界面，也不值得花钱去购买。而R提供基础的地图功能，而且还免费，但可能达不到你想要的效果。如果你希望创建在线的、可交互的地图，可以选择开源的Modest Maps或Polymaps，但它们又需要更高的编程技能。在第8章你将会学到如何使用各种可用的工具。

3.5 衡量各种可选项

以上列出的数据可视化工具也许并不完全，但用它们作为起步应该足够了。不过我们还是需要考虑和权衡。使用何种工具主要取决于你想完成的目标，而且同一个任务会有多种完成途径，甚至在同一个软件中都是如此。想设计静态数据图？不妨试一下R或者Illustrator。想为某个Web应用创建可交互图表？那么请试试JavaScript或者Flash。

在FlowingData网站上，我发起了一次调查，询问人们主要使用什么来分析和可视化数据。有超过1000人进行了回应，其结果如图3-29所示。

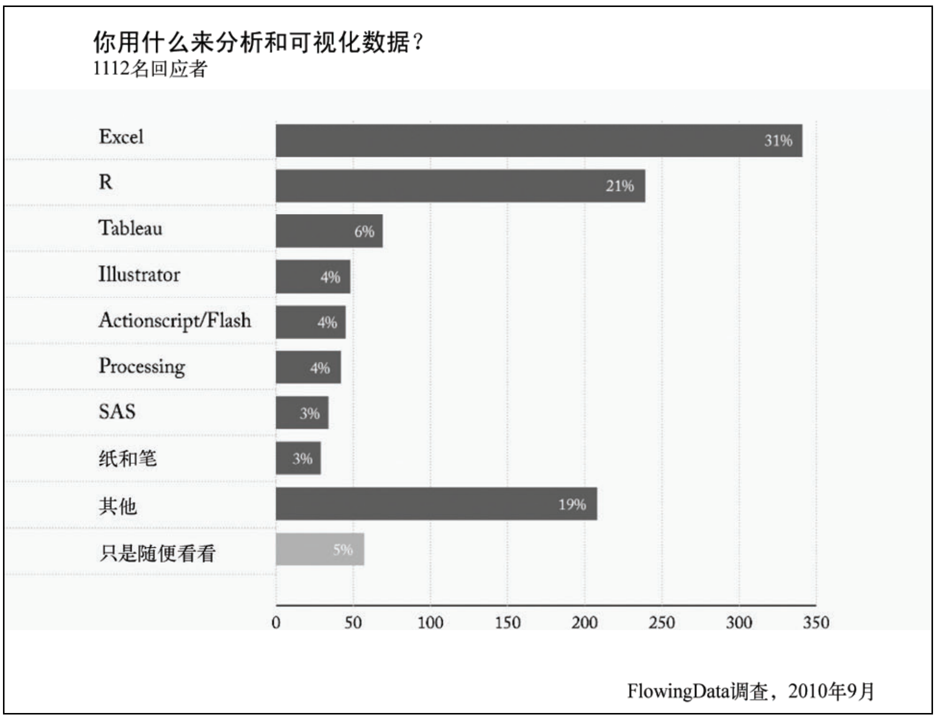


图3-29 FlowingData的访问者都用什么来分析和可视化数据

我们可以看到有几个明显的领先者。Excel排名第一，而R紧随其后。但此后软件的选择开始分化。有超过200人选择了“其他”类别。在评论中，很多人表示他们使用多款工具来满足不同的需求，从长期来看这也是最好的途径。

混合使用

很多人都喜欢坚持用一款软件——这样更舒适轻松，不用学任何新的东西。如果这真的有效的话，他们当然可以继续下去。但当我们与数据打交道足够久之后，就会遭遇软件的瓶颈。你知道自己希望实现的效果，但软件却不允许你这么做，或者给你制造一些不必要的麻烦。

你要么接受现实，要么使用别的软件，虽然需要时间去学习，但却有助于实现你的愿景——我会选择后者。掌握多种工具可以确保你不会陷入僵局，而且让你更加多才多艺，轻松应对各种可视化任务并得到满意的结果。

3.6　小结

请记住，在这些工具中没有哪一个是万用良方。归根结底，对数据的分析和设计取决于你自己。工具仅只是工具而已，你手中拿着锤子并不代表着你就能建造高楼大厦。即使你的计算机再强劲，安装了再强大的软件，如果你不知道如何运用这些工具，它们就没有任何价值。先想好要提出什么问题、使用何种数据、强调哪些方面，那么在真正实践的过程中一切就会容易许多。

不过你是幸运的，因为我们马上就会谈到这些。后续章节将会涵盖那些重要的数据设计概念，并教大家如何将抽象的理论付诸实践（通过上述的各种工具）。你会学到从数据中找寻亮点并实现可视化的方法。

有关时间趋势的可视化 4

时间序列数据无处不在。舆论在变化，人口在迁移，企业在发展。我们通过时间序列数据来观察这些事物是如何变化的。本章会分别讨论离散时间和延续时间的数据，因为不同的数据类型决定了图表的表现形式。我们还会亲身体验R和Adobe Illustrator的强大功能——结合使用这两款软件实现的效果简直太棒了。

4.1　在时间中寻求什么

我们每天都在看时间。它在电脑里，手表上，手机中，以及其他能看到的任何地方。即使没有钟表，我们也会从日常起居和日升日落中感受到时间。获得有关时间的数据是再自然不过的事情，它让我们了解到事物正在如何变化。

不管是延续性还是暂时性的时间数据，我们最常想的是从中发现趋势。某个东西是在上升还是下降？是否存在周期性的循环？要想找出这些变化中存在的模式，就必须超越单个数据点，纵观全局。只观察某个时间点上的数值当然更轻松，但只有在了解到来龙去脉之后，你才会对这个数值产生更深刻的理解。而且你对数据了解得越多，你所讲的故事就越有感染力。

比如说，奥巴马政府在总统就职一年后公布了一份图表，我在图4-1中进行了重新绘制。它显示了从布什卸任到奥巴马上任之初这段时间内的失业情况。

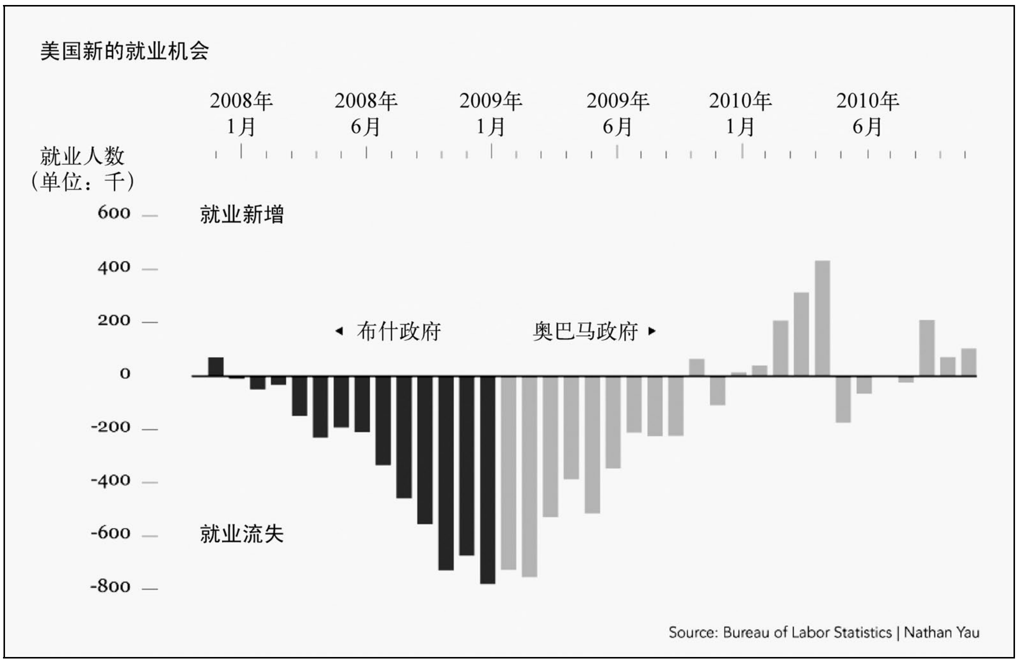


图4-1　巴拉克·奥巴马执政后失业的变化

从上图看来，似乎新政府对失业产生了显著的积极效果。但如果把当前图表缩小，以更长的时间范围来观察呢（见图4-2）？你是否会得出不一样的结论？

尽管很多人都希望能了解全局，但关注细节也同样很有价值。有没有数值存在异常？有没有不规则的时间段？有没有呈现出剧烈的上升或下降？如果有，是什么原因导致的？一般来说，这些不符常规的地方正是我们应该关注之处。当然它也可能是因为数据条目中出现了错误。放眼全局，了解事件的整个背景，这能帮助你甄别它到底属于何种情况。

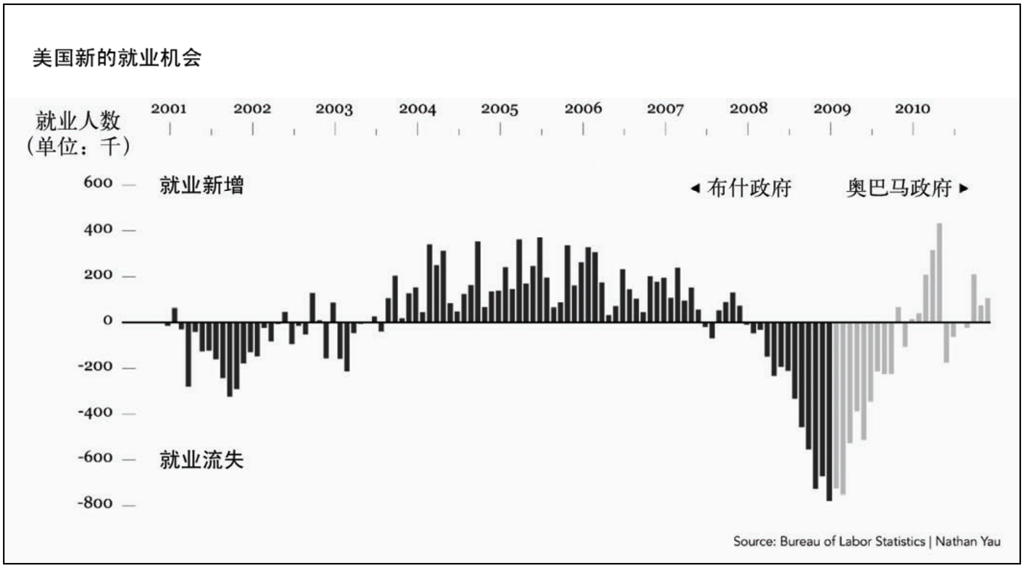


图4-2 2001—2010年的失业变化

4.2 时间中的离散点

时间数据可以分为离散时间和延续时间两种。理解自己的数据属于何种类型，你才能选择正确的可视化方法。在离散型时间数据中，数据来自于某个具体的时间点或者时段，可能的数值也是有限的。比如说，人们每年的考试通过率就是离散型数据。人们去考试，然后事件就结束了。考试有具体的日期，而且人们的分数也不会再发生变化。而类似温度这样的数据则是延续型的，在一天当中的任何时刻都可以去测量它，而且它一直都在变化。

在本节中你将会看到适用于可视化离散型时间数据的图表类型，并且通过实例学习如何在R和Illustrator中创建这些图表。在每种类型的一开始我都会作简要介绍，其中涉及的设计模式将贯穿本章，因此这些部分非常重要。虽然实例针对的都是具体图表，但其设计原则是通用的，所有类型的可视化都可以借鉴。不要忘了，要建立大局意识。

4.2.1 柱形

柱形图是最常见的图表类型，你可能见过不少，甚至还绘制过一些。柱形图适用于多种类型的数据，我们先看看在时间数据中如何使用它。

图4-3显示了基本的框架。时间轴（水平轴，也就是*x*轴）显示了按先后顺序排列的各时间点。本图中显示的时间点是月份，从2011年1月到6月，你也可以显示年份、日期或其他时间单位。柱形的宽度和彼此之间的间隔一般不代表数值。

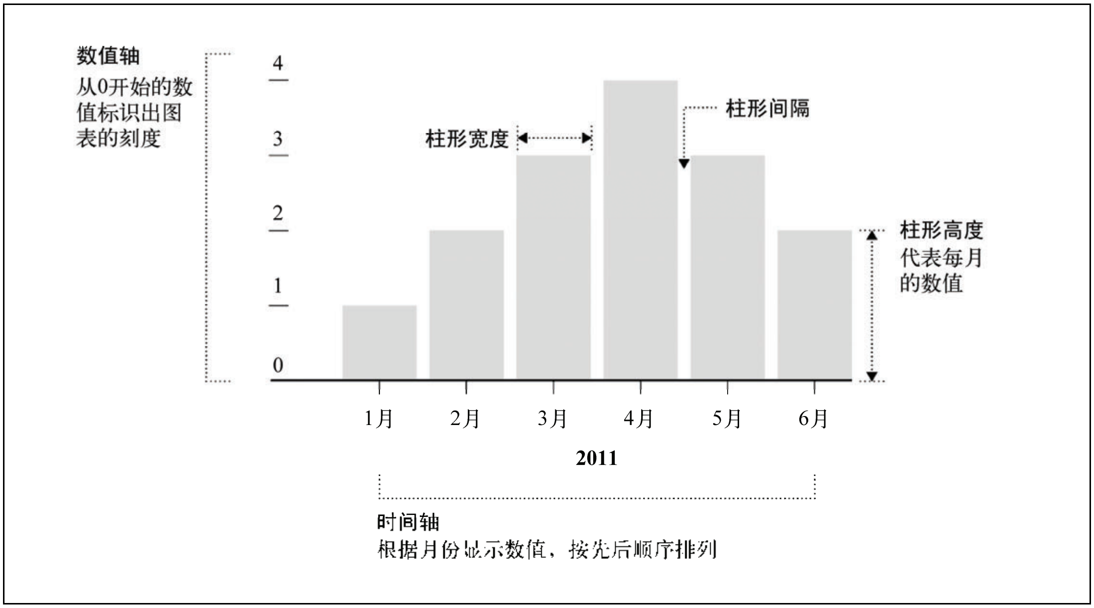


图4-3　柱形图的基本框架

数值轴（垂直轴，也就是*y*轴）标识出图表的尺度。图4-3显示的是线性标尺（linear scale），各单位在整条轴线上均匀分布。柱形高度与数值轴相对应。比如说第一个柱形达到了1个单位，而最高的柱形达到了4个单位。

这很重要。体现数值的视觉线索就是柱形的高度。数值越小，柱形就越矮；数值越大，柱形就越高。所以我们可以看到4个单位的柱形（4月）的高度是2个单位的柱形（2月）的两倍。

许多程序默认将数值轴上的最小值设为数据集中的最小值，如图4-4所示。本例中的最小值是1。然而，如果数值轴从1开始，那么2月柱形的高度就不再是4月柱形高度的一半了。看上去2月是4月的1/3。而且1月的柱形也消失了。所以请永远保持数值轴尺度从0开始标注。否则你的柱形图就可能会显示出错误的关系。

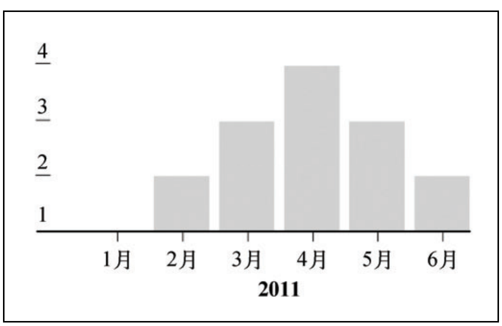


图4-4　数值轴不从0开始的柱形图

提示　处理的数据都是正数时，请永远让柱形图的数值轴从0开始，否则会让人们难以从视觉上比较各柱形的高度。

1. 创建柱形图

是时候制作你的第一张图表了。我们使用的数据是真实的，来自人类历史上最重要的赛事近30年的比赛成绩。这一赛事绝对值得被每一个人铭记——它就是享誉国际的“内森杯”热狗大胃王比赛！

图4-5是最终生成的图表。我们会分两步完成。首先在R中创建基本柱形图，然后在Illustrator中进行完善。

如果你还不知道这项巡回赛事，不妨稍微了解一下：“内森杯”热狗大胃王比赛每年举办一次，日期是在7月4日，也就是美国独立日。它实在太受欢迎了，连ESPN电视台都会进行直播。

在20世纪末期，获胜者能在约15分钟内吃下10～20个热狗面包（hot dogs and buns，HDB）。然而在2001年，来自日本的专业级食客小林尊（Takeru Kobayashi）以50个HDB的成绩傲视群雄，比他之前的世界冠军超出了整整一倍还要多。直到这时故事才开始进入高潮。

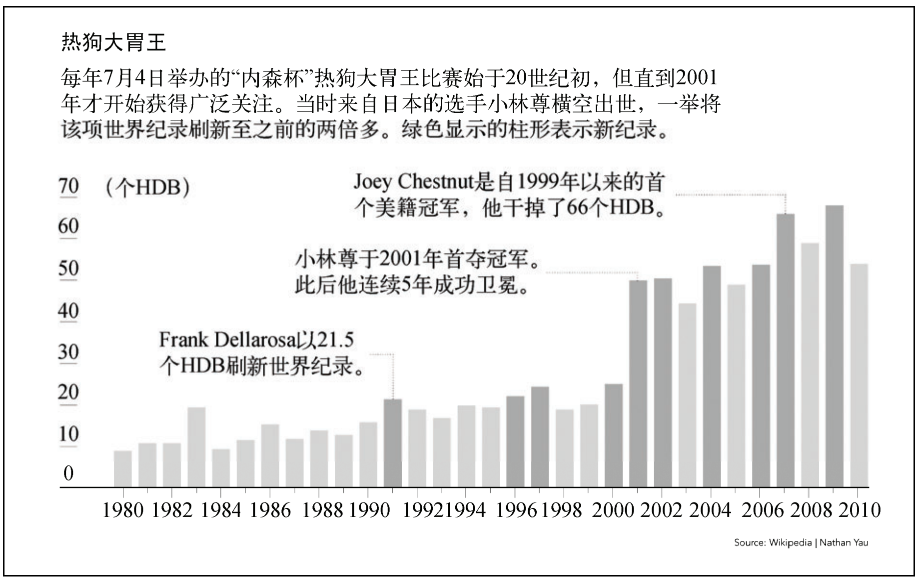


图4-5　显示“内森杯”热狗大胃王比赛成绩的柱形图

维基百科提供了这项赛事从1916年至今的所有成绩，但大胃王比赛从1980年开始才成为一项常规赛事，所以我们可以从这里开始。数据是以HTML表格形式呈现的，包括了赛事年度、选手姓名、吃掉的HDB数量和获胜者的国籍。我已经把数据编辑成了CSV文件，大家可以到http://datasets.flowingdata.com/hot-dog-contest-winners.csv下载。数据的开头五行是这样的：

```
"Year","Winner","Dogs eaten","Country","New record"
1980,"Paul Siederman & Joe Baldini",9.1,"United States",0
1981,"Thomas DeBerry ",11,"United States",0
1982,"Steven Abrams ",11,"United States",0
1983,"Luis Llamas ",19.5,"Mexico",1
1984,"Birgit Felden ",9.5,"Germany",0
```

▶访问http://datasets.flowingdata.com/hot-dog-contest-winners.csv下载CSV格式的数据。在维基百科网站可以浏览“内森杯”热狗大胃王比赛的历史以及未经编辑的数据。

使用read.csv()命令在R中载入数据。你可以从自己电脑上本地载入CSV文件，也可以通过URL加载。要想实现后者，可以在R中输入下面这行代码：

```
hotdogs <-
    read.csv("http://datasets.flowingdata.com/hot-dog-contest-winners.csv",
    sep=",", header=TRUE)
```

如果你打算从自己电脑上本地载入数据文件，请通过主菜单把R的工作路径设为你的数据文件所在的文件夹：点击File（文件）菜单下的Change dir...（修改目录）选项。当然你也可以使用setwd()函数来设置。

如果你是不熟悉编程，可能会难以理解以上代码，我们把它拆开来讲解。这是一行R代码，目的是通过read.csv()命令载入数据，其中有3个参数。第一个是数据的来源位置，本例中是一个URL地址。

第二个参数sep指定了数据文件中分隔各列的字符是什么。这是一个逗号分隔文件，所以将它指定为逗号。如果是一个制表符分隔的文件，就需要用\t来代替逗号，表明分隔符是制表符。

最后一个参数header告诉R数据文件是有标头的，它包含有每一列的名称。第一列是赛事年度，第二列是获胜者姓名，第三列是吃掉的HDB数量，第四列是该获胜者的国籍。你可能注意到我还添加了一个新的字段new record。如果该年度的世界纪录被打破，那么值就取1；否则就是0。我们很快就会用到它。

现在数据已经导入到R，可以通过hotdogs变量来读取。技术上来说，数据其实是被存储为一个数据块（data frame），这一点并不重要，但从概念来说是如此。如果键入hotdogs就会看到类似如下的数据块（开头部分）。

```
  Year                      Winner Dogs.eaten       Country New.record
1 1980 Paul Siederman & Joe Baldini       9.10 United States          0
2 1981              Thomas DeBerry      11.00 United States          0
3 1982               Steven Abrams      11.00 United States          0
4 1983                 Luis Llamas      19.50        Mexico          1
5 1984               Birgit Felden       9.50       Germany          0
```

列名称中的空格已经被替换为句点。Dogs eaten现在变成了Dogs.eaten，New record也是如此。要访问具体某一列数据，你需要在数据块名称后面加上美元符号（$），然后再加上该列

的名称。比如要访问Dogs.eaten列，你需要输入：

```
hotdogs$Dogs.eaten
```

现在R中已经有了数据，可以直接通过barplot()命令来绘图了。

```
barplot(hotdogs$Dogs.eaten)
```

这行代码告诉R绘制出Dogs.eaten列，应该会得到图4-6。

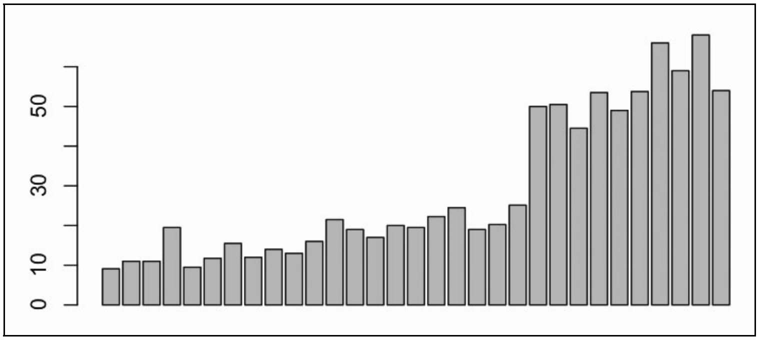


图4-6 通过R的barplot()命令默认生成的热狗数量图表

不赖，但你还能做得更好。通过在barplot()里设置names.arg参数来指定每一个柱形的名称。在本例中也就是每次赛事的举办年份。

```
barplot(hotdogs$Dogs.eaten, names.arg=hotdogs$Year)
```

这样就能得到图4-7。图表的底部出现了标记。

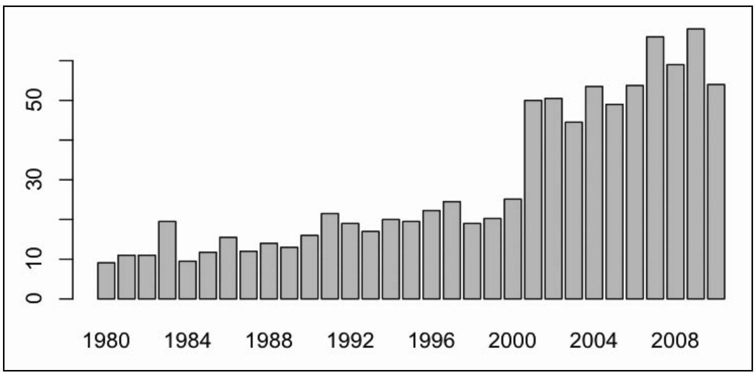


图4-7 带有年份标记的柱形图

还有很多其他参数可以设置。你可以添加坐标轴标签、修改边框和颜色，如图4-8所示。

```
barplot(hotdogs$Dogs.eaten, names.arg=hotdogs$Year, col="red",
    border=NA, xlab="年份", ylab="吃掉的HDB数量")
```

其中的col参数可以让你选择颜色（在R的说明文档中有标明），或者采用16进制的色彩编码，

例如`#821122`。同时你用`NA`指定了去除边框，这是一个逻辑常数，代表没有数值。此外你还分别以“年份”和“吃掉HDB数量”标记了*x*轴和*y*轴。

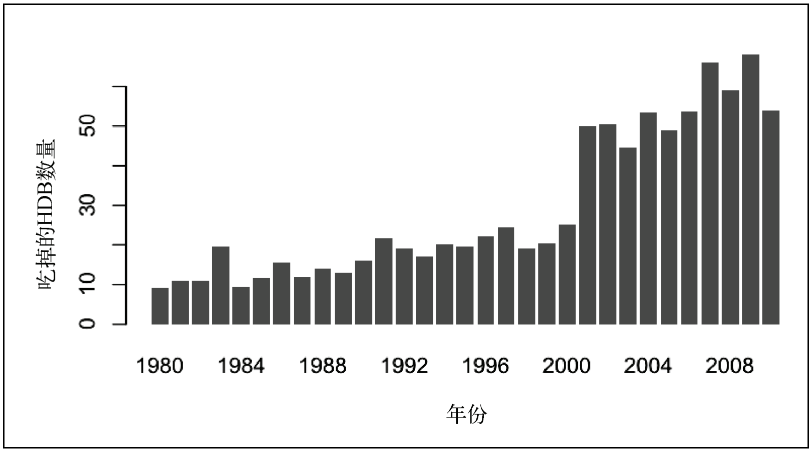


图4-8　设置颜色及坐标轴标记后的柱形图

并不是只能用一种颜色。通过为`barplot()`提供多种颜色，你可以给想要的柱形着色。比如说你想突出显示美国人获胜的年份，那么就把该年度的柱形显示为暗红色（`#821122`），而让其他柱形显示为浅灰色，如图4-9所示。

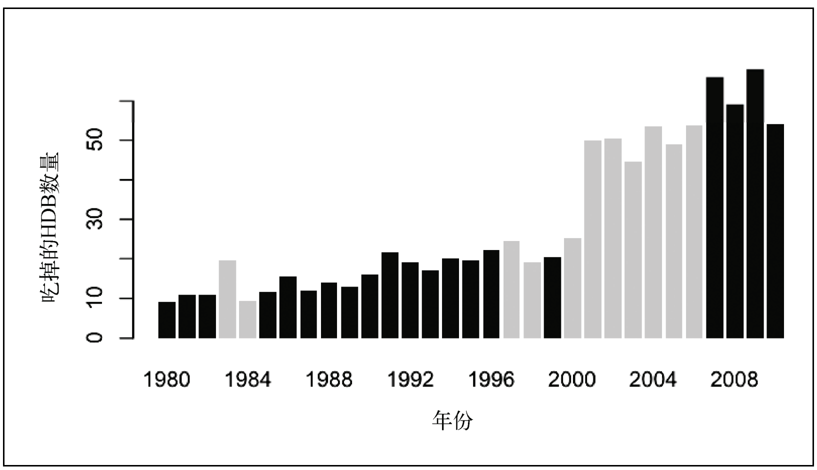


图4-9　柱形呈现多种色彩的柱形图

要实现这一目的，你需要创建一个列表（也就是R里面的向量）用于着色。访问每一年的数据然后决定相应的柱形应该采用哪种颜色。如果是美国人获胜，指定为暗红色；否则指定为灰色。以下是相关代码：

```
fill_colors <- c()
for ( i in 1:length(hotdogs$Country) ) {
    if (hotdogs$Country[i] == "United States") {
        fill_colors <- c(fill_colors, "#821122")
    } else {
        fill_colors <- c(fill_colors, "#cccccc")
    }
}
```

第一行启用了一个名为fill_colors的空向量（vector）。我们用c()来在R里创建向量。

第二行开始了一个for循环。代码告诉R在名为i的索引中进行循环，其中i从1开始，最高值是hotdogs数据块的所有行数。具体来讲，你从hotdogs数据块中取得一个列，本例中是Country列，然后找到它的长度。如果只是对hotdogs用length()，就只能得到它的列数，也就是5。但我们要的是它的行数，应该是31。1980—2010年的每一年都占一行，所以循环会在括号中执行31遍代码，其中每循环一次，索引i都会加1。

所以在第一次循环中，也就是i等于1时，R会检查第一行（也就是1980年）的获胜国是否为美国。如果是，就在fill_colors中添加颜色#821122（一种暗红色，以16进制形式显示）。如果不是，则添加颜色#cccccc，也就是浅灰色。

4

说明　大多数编程语言的数组或向量都以0开始，也就是索引的第一项从0开始引用。不过R中的向量是从1开始的。

1980年的获胜者来自美国，所以我们执行前者。之后循环会检查余下的30个年份。在R的控制台中键入fill_colors可以看到程序运行的结果。这是一组按我们需要建立的色彩值向量。

以下代码可以把fill_colors这组向量传递到barplot()的col参数中去，以便显示更新后的柱形图。

```
barplot(hotdogs$Dogs.eaten, names.arg=hotdogs$Year, col=fill_colors,
    border=NA, xlab="Year", ylab="Hot dogs and buns (HDB) eaten")
```

这段代码和之前的基本一样，只不过在col参数中用fill_colors代替了之前的"red"。

不过，图4-5中的最终图表突出显示的是纪录被刷新的年份，而不是美国获胜的年份。没关系，整个过程和逻辑思路是一样的。你只需修改一些条件即可。在数据块中的New.record列代表的是纪录是否被刷新，所以如果它是1，就使用暗红色，否则使用灰色。以下是R中实现的代码。

```
fill_colors <- c()
for ( i in 1:length(hotdogs$New.record) ) {
    if (hotdogs$New.record[i] == 1) {
        fill_colors <- c(fill_colors, "#821122")
    } else {
        fill_colors <- c(fill_colors, "#cccccc")
```

```
    }
}
barplot(hotdogs$Dogs.eaten, names.arg=hotdogs$Year, col=fill_colors,
    border=NA, xlab="Year", ylab="Hot dogs and buns (HDB) eaten")
```

代码与美国获胜的例子基本一样，但在if语句中略有变化。结果应该如图4-10所示。

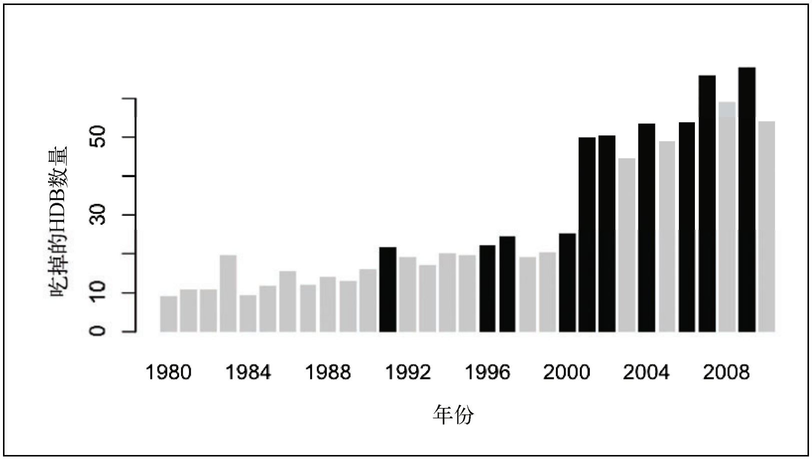


图4-10　柱形呈现多种色彩的柱形图，不过运用的条件和图4-9不同

现在你可以试一下barplot()里的其他选项，例如柱形间隔或者添加标题。

```
barplot(hotdogs$Dogs.eaten, names.arg=hotdogs$Year, col=fill_colors,
    border=NA, space=0.3, xlab="Year", ylab="Hot dogs and buns (HDB)
eaten")
```

这样你就能得到图4-11。注意柱形间隔较之前更宽，而且顶部也出现了标题。

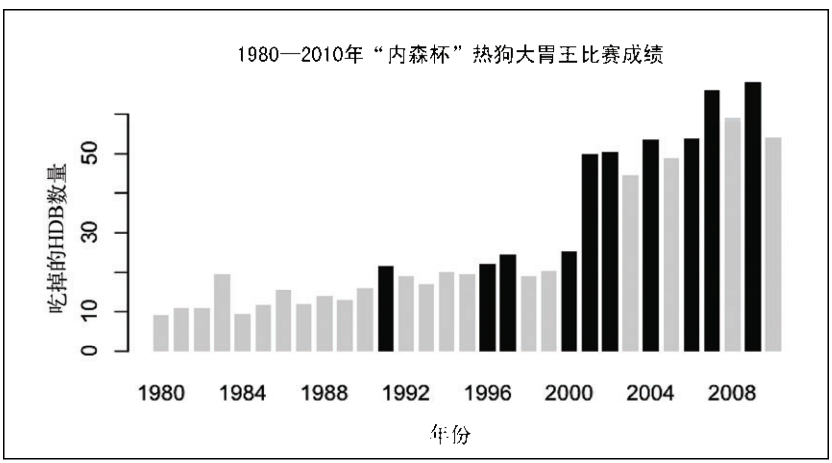


图4-11　带有自定义间隔和主标题的柱形图

提示　在选择柱形间隔时需要小心。如果间隔的宽度和柱形宽度相近，视觉上就会产生振动效应（vibration effect），给人感觉柱形和间隔的角色发生了互换。

嘿！你的第一次R尝试已经成功完成了！

先选中图表的显示窗口，然后在File（文件）菜单中选择保存图表。将图表保存为PDF文件，你马上就会用到它。

提示　要想查阅R中函数的说明文档，只需要输入一个英文问号再加上函数名称即可。比如要在R中查阅`barplot()`，只需输入`?barplot`，R就会提供该函数的描述以及所有可用参数的解释说明。通常还会有可运行的实例，非常有帮助。

---

2. 在Illustrator中完善图表

现在你有了一张基础的柱形图，看起来还不赖。如果你只用它来方便自己分析，那么就没必要再进行加工了。但如果你希望把它制成可单独发布的图表，就得再花点工夫让它更具可读性。

让我们从讲故事的角度来看。假设你是一位读者，无意中看到了图4-11这张尚未完善的图表，你从中能获得哪些信息？你能看出它显示的是每年被吃掉的热狗面包。它是否记录的是某个人的饮食习惯？很明显对普通人来说这些热狗也太多了点。或者是给动物吃的？吃剩的会拿去喂鸟吗？或者是每个人每年平均食用的热狗数量？为什么有些柱形的颜色还不一样呢？

作为绘制图表的人，你了解数字背后的上下文背景，但是你的读者不了解。所以你必须对正在发生什么作出解释。好的数据设计可以帮助读者更清楚地理解整个故事。Illustrator可以帮助你做到这一点。通过它你能直接修改图中的每一个元素，比如调整字体、添加说明、修改坐标轴、编辑颜色，完成你能想象到的任何效果。

提示　站在读者的角度来设计数据图表。有哪些部分需要加以解释？

提示　如果你没有Illustrator，可以尝试InkScape，它是免费且开源的替代工具。虽然它在功能或界面上和Illustrator不太一样，但仍然可以从中找到许多共同点。

本书只是用Illustrator对图表稍作修改。但在学习了更多实例，并且开始自己设计图形时，你就会发现这些小小的改动能极大地改进图表，让它们更加简洁明了。

让我们先走好第一步。在Illustrator里打开刚才柱形图的PDF文件。你应该会在一个窗口中看到自己制作的图表，此外还有其他一些小面板提供了各种工具，包括颜色和字体面板。最值得注意的是Tools（工具）面板，如图4-12所示。你会经常用到它。如果你在界面中找不到Tools面板，可以在Window（窗口）菜单中单击Tools选项来打开它。



图4-12　Illustrator的Tools面板

面板上的黑色箭头图标被称为“选取工具”（Selection tool）。选中它，你的鼠标指针就会变成一个黑色箭头。单击并拖动鼠标，选择图表的整个边框。此时边框会高亮显示，如图4-13所示。它在Illustrator中被称为“剪切蒙版”（clipping mask）[①]。在许多情况下这个东西都很有用，但现在你不用在意它。单击Delete键把它删除。如果这个操作删除了整个图表，撤销本次操作，然后使用直接选取工具（Direct Selection tool），也就是白色箭头图标，它能直接选中剪切蒙版。

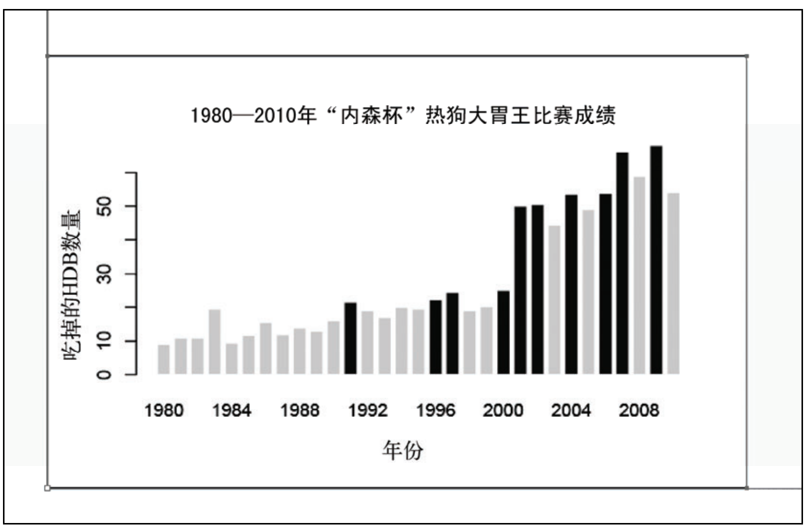


图4-13　删除PDF的剪切蒙版

① 在译者使用的Adobe Illustrator CS5版本中，打开R输出的PDF后似乎并未出现剪切蒙版，可以直接选择图表。

现在尝试改变字体，这个操作很简单。再次使用选择工具，单击你想编辑的文本来选中它。在“字体”（Font）面板中通过下拉菜单选择你喜欢的字体，如图4-14所示。你也可以通过“文字”（Type）菜单来改变字体。图4-15中的字体已经被改成了Georgia Regular。



图4-14 Illustrator中的“字体”面板

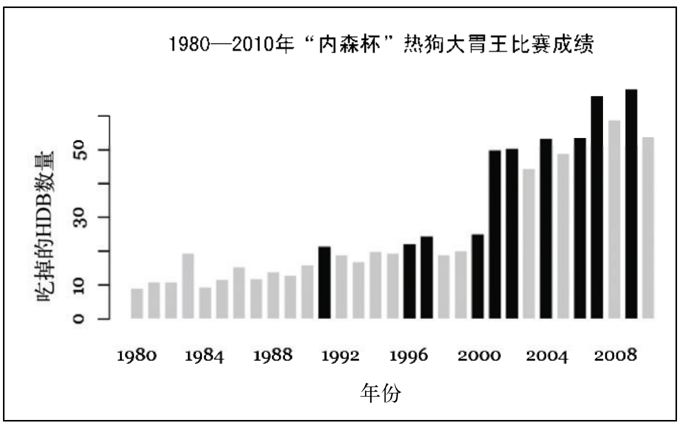


图4-15 字体改为Georgia Regular后的图表

下一步，数值轴上的那些数字标签也需要调整。现在的数字都是侧立的，应该将其正过来以提高可读性。单击那些数字，你会注意到其他元素也会被选中。这是因为数字和其他元素默认属于一个群组[①]。你需要先打散群组才能旋转各个数字，可以在Object（对象）菜单中找到Ungroup（取消编组）选项并点击。取消已选定的数字标签，然后再选择它们，此时不再是选中整个组了，只有数字会被选中。有的时候你可能需要打散好几次群组才能只选中想要的对象。或者你也可以通过直接选择工具来选取想要的元素。

选中各个数字后，回到“Object”菜单，选择Transform （变换）→Transform Each（分别变换）。在图4-16所示的面板中，将旋转角度设置为–90°。然后单击“确定”（OK）按钮。现在数

① 在Adobe Illustrator CS5版本中，打开R输出的PDF后似乎并未出现群组，可以直接选择图表。

字标签就都正过来了。

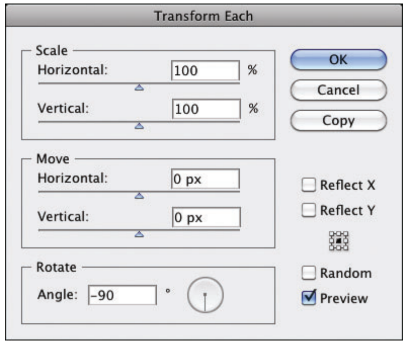


图4-16　Transform Each面板

保持现在的选中状态不变，将数字标签往上、右方移动（不包括刻度线），让它们从刻度线的左侧移动到刻度线的上方。你可以用键盘上的方向键来移动对象，或者直接用鼠标拖动它们。此外还可以修改*y*轴左侧的文本，并将其移动到*y*轴的顶部右侧。这样也能提高可读性，因为读者的视线一般会从左到右移动。

为了让图表更加简洁，你还可以去掉数值轴上的垂直线。它对于数据传递并没有多大帮助。请留意在图4-5的最终图表中只有刻度线。如果你用选择工具点击垂直线，有可能那些标签也会被选中，这是因为它们默认在一个群组内。可以用直接选择工具选中这根线，然后敲Delete键，它就会消失。现在你的图表应该和图4-17类似。

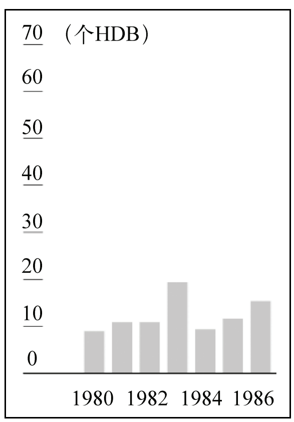


图4-17　简化数值轴后的柱形图

提示　数据图的目的是显示数据本身。尝试去掉所有多余的元素。

现在你的图表已经和图4-5更加接近了。不过它依然缺少一些东西。在水平轴上没有任何刻度线、图表也缺少注释，而且如果图表中包含了绿色会更好，因为绿色是“内森杯”热狗大赛的标志色之一。

创建刻度线有很多种方法，比如说通过钢笔（Pen）工具，你可以轻而易举地画出一条直线。在“工具”面板中选择该图标，然后在“描边”（Stroke）面板中指定线条的样式。将笔触粗细设置为0.3 pt，并确定“虚线”（Dashed Line）复选框没有被选中。

要画一根线，先在你希望的地方（比如第一个柱形的正下方）单击鼠标确定线段的一端，然后在稍下方再次单击鼠标确定另一端。如果在第二次单击时按住Shift键，线段就会自动呈现垂直或水平状态。现在你应该画出了第一根刻度线。你还需要30条这样的线段，因为每一年都需要一个刻度。

你可以一根一根把它们都画出来，但还有更好的办法。如果你用的是Mac，按住Option键；如果你用的是PC，则按住Alt键。用选择工具单击之前画出的刻度线然后拖动到你希望下一根刻度线所在的位置（比如第二个柱形的正下方，稍有偏移也没关系）。在这个过程中要保持按住Option或Alt键，然后松开鼠标。这个操作会创建一个副本，所以你现在应该有两根刻度线了。现在松开Option或Alt键，再按一下Command+D（Mac）或Ctrl+D组合键（PC）。这个操作会复制一根新的刻度线，而且它与第二根线的距离与前两根线的间距相等。持续按Command/Ctrl+D组合键，直到获得足够多的刻度线为止。

最后，还要适当排列一下所有的刻度线。将最后一根刻度线移动到最后一个柱形的正下方。第一根刻度线应该已经位于第一个柱形的正下方了。现在选中所有的刻度线，然后在“对齐”（Align）面板（见图4-18）中单击“水平居中分布”（Horizontal Distribute Center）图标。

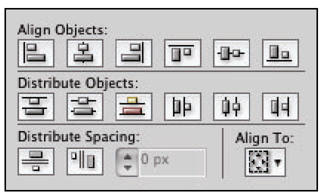


图4-18　Illustrator中的“对齐”面板

这个操作会重新分配每根刻度线的间距，使它们均匀分布在首末两根刻度线之间。你还可以用选择工具选中偶数位的刻度线，然后缩短它们的垂直长度。这样能更明显地表现出稍长的刻度线对应了已有的年度标签。

要想把填充色从红色改为绿色，可以用直接选择工具选中每一个红色矩形。由于这些矩形数量不多，选起来还比较快。但如果有很多矩形需要选中怎么办？你可以先选中一个红色柱形，然后在菜单中单击“选择”（Select）→“相同”（Same）→“填充颜色”（Fill Color）。这个操作能满足你的要求，选中所有以红色填充的元素。现在只需在“颜色”（Color）面板中设置你想要的颜色即可。你可以在这里改变边框色和填充色，不过目前只需改动填充色就行了，效果如图4-19所示。

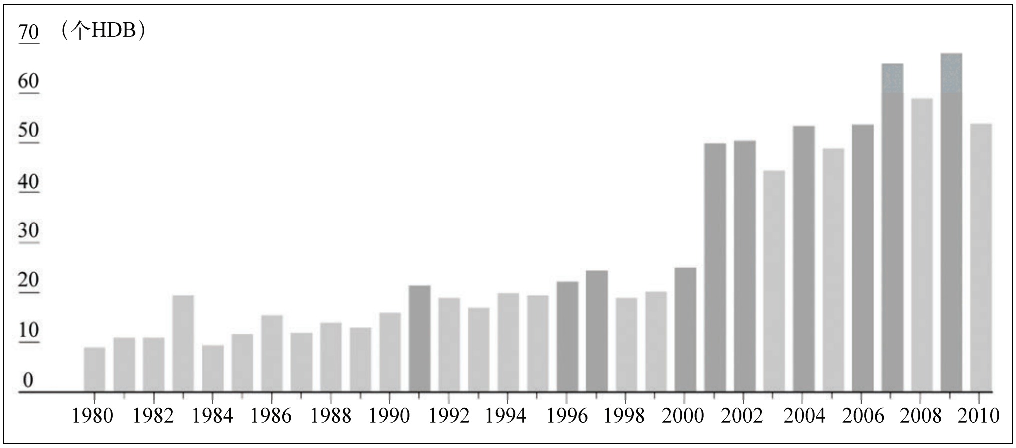


图4-19　改变图表元素的颜色

在“工具”面板中找到文字（Type）工具，可以在图表中添加文本。这是一个给图表加注释、阐明所有疑点的好机会。选择你想要的字体、字号和样式，以便和图表中其他标记文字（例如坐标轴标签）区分开来。

在这个重要非凡的热狗图表中，有3个年度可以着重强调，包括1980年后的第一次纪录刷新、小林尊的统治伊始和Joey Chestnut代表美国夺魁。同时还需要添加图表标题和说明，以便点明整幅图表的要旨。

最后也是相当重要的一点：别忘了添加数据来源。除此之外没办法证明你的图表的准确性。

提示　确保在你的图表中提供了数据来源。它不仅提升了可信度，而且还提供了上下文背景。

所有因素加在一起，你就得到了最终的图表，和图4-5中的一样。

你可能需要一段时间来消化这些内容，但当你和图表打的交道越多时，这一切就会变得越简单。如果你理解了刚才的所有操作，就能发现用R或其他任何语言编写代码所遵循的模式。而且虽然Illustrator博大精深，你也已经掌握了其中一些常用的功能。

后续的例子会涉及有关时间数据的其他图表类型，而且会继续用到R和Illustrator。鉴于你已经了解了这两款软件的基础知识，后面的进度将略有加快。

4.2.2　柱形的堆叠

如图4-20所示，堆叠柱形图的几何形状与常规柱形图很相似，区别自然在于矩形都彼此堆叠。如果数据存在子分类，且各子分类之和有意义，我们就可以用到堆叠柱形图。

和柱形图一样，堆叠柱形图可以用于多种数据类型，并不只针对时间数据。不过在图4-19的例子中，*x*轴是按月显示的。

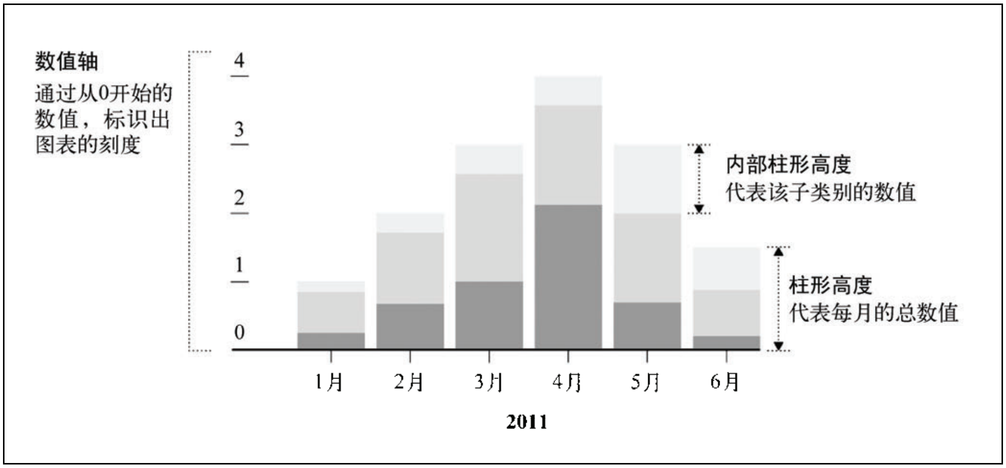


图4-20　堆叠柱形图的基本框架

创建堆叠柱形图

堆叠柱形图也是很常见的图表类型，有多种制作方法。我们的例子是用R来实现的。制作过程和创建常规柱形图很相似。

(1) 载入数据。

(2) 确保数据有适当的格式。

(3) 通过某个R函数来创建图形。

每次用R制作数据图表你都会经过这三个步骤。有时你可能会在某个步骤中花费更多的时间。也许是要琢磨数据的格式，也许是必须自己写函数才能实现某种效果。但不管怎样，你总会经过以上三个步骤，而且在其他语言中也是如此，后面几章中你就会见到。

回到堆叠柱形图，让我们再看看“内森杯”热狗大胃王比赛。我们后面不会再看到有关热狗的数据，所以请尽情享受这一刻。图4-21是我们希望实现的效果。

这一次，我们不再只是观察赛事冠军吃掉的热狗数量，而是扩大到每年的前三甲选手。每一个堆叠都代表一个年份，包含三个柱形，每一个柱形代表一位前三甲选手。维基百科从2000年才有相关的数据记录，所以我们也由此开始。

让我们先走好第一步，在R中载入数据。你可以直接从以下URL获得数据。

```
hot_dog_places <-
    read.csv('http://datasets.flowingdata.com/hot-dog-places.csv',
    sep=",", header=TRUE)
```

输入`hot_dog_places`查看数据。每一列显示了该年度的比赛结果，每一行分别表示冠、亚、季军。

```
  X2000 X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009 X2010
1    25  50.0  50.5  44.5  53.5    49    54    66    59  68.0    54
2    24  31.0  26.0  30.5  38.0    37    52    63    59  64.5    43
3    22  23.5  25.5  29.5  32.0    32    37    49    42  55.0    37
```

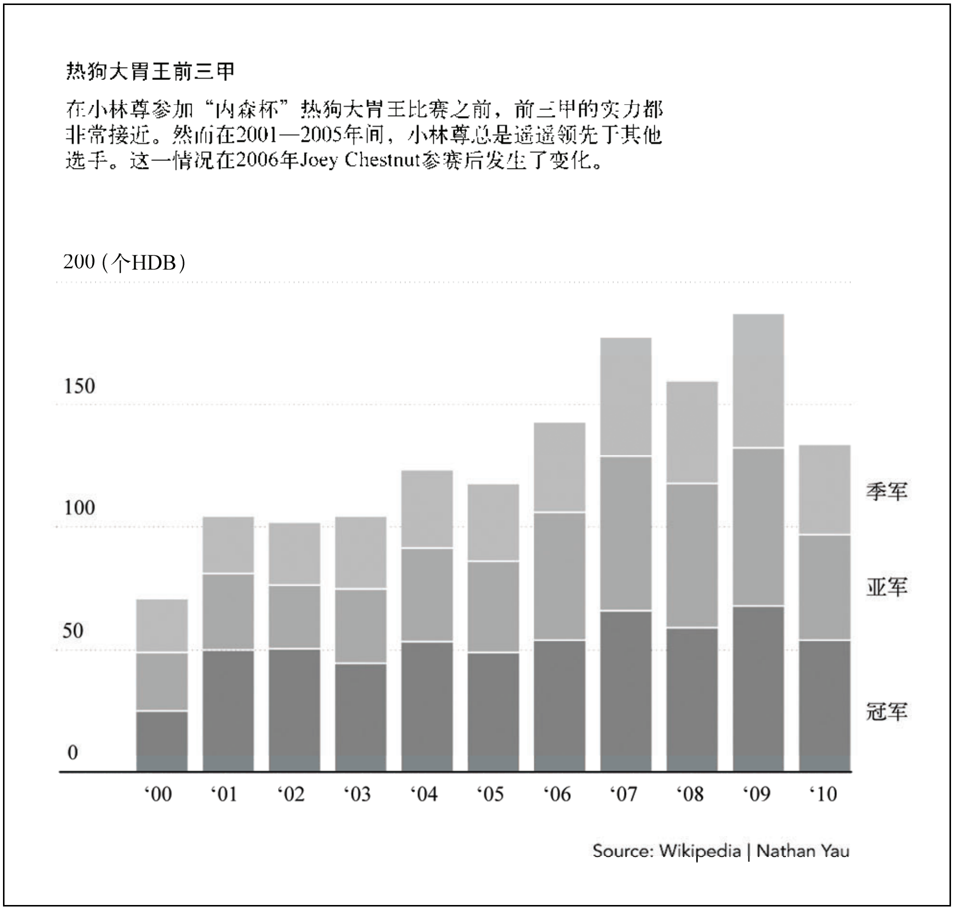


图4-21　显示2000—2010年各年度大胃王前三甲的堆叠柱形图

所有的列名称都以“X”开头，这是R在载入数据时自动添加的，因为R不喜欢标头的名称是数字，所以它强加了一个字母让它们更像一个单词（用更专业的词汇来说就是字符串）。最终图表里的坐标轴标签会显示这些列名称，所以你需要把它们改回来。

```
names(hot_dog_places) <- c("2000", "2001", "2002", "2003", "2004",
    "2005", "2006", "2007", "2008", "2009", "2010")
```

把每个年份用双引号括起来，用于指明它是一个字符串。再次输入`hot_dog_places`，标头现在只剩下年份了。

```
  2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
1   25 50.0 50.5 44.5 53.5   49   54   66   59 68.0   54
2   24 31.0 26.0 30.5 38.0   37   52   63   59 64.5   43
3   22 23.5 25.5 29.5 32.0   32   37   49   42 55.0   37
```

和之前一样，我们将用到barplot()函数，但会是另一种格式。要想把上面的所有数值都传递给barplot()，你需要把hot_dog_places转换成一个矩阵（matrix）。现在它还只是一个数据块。这两者在R中的结构是不同的，不过其中的区别就目前来说并不重要。你只需要知道如何将数据块转换成矩阵就行了。

```
hot_dog_matrix <- as.matrix(hot_dog_places)
```

你已经把我们新创建的矩阵存储为hot_dog_matrix。现在可以把它传递给barplot()。

```
barplot(hot_dog_matrix, border=NA, space=0.25, ylim=c(0, 200),
xlab="Year", ylab="Hot dogs and buns (HDBs) eaten",
main="Hot Dog Eating Contest Results, 2000-2010")
```

你指定了柱形没有边框、间距是柱形宽度的0.25倍、数值轴的范围是0～200，并加上了图表的标题和坐标轴标签。结果如图4-22所示。

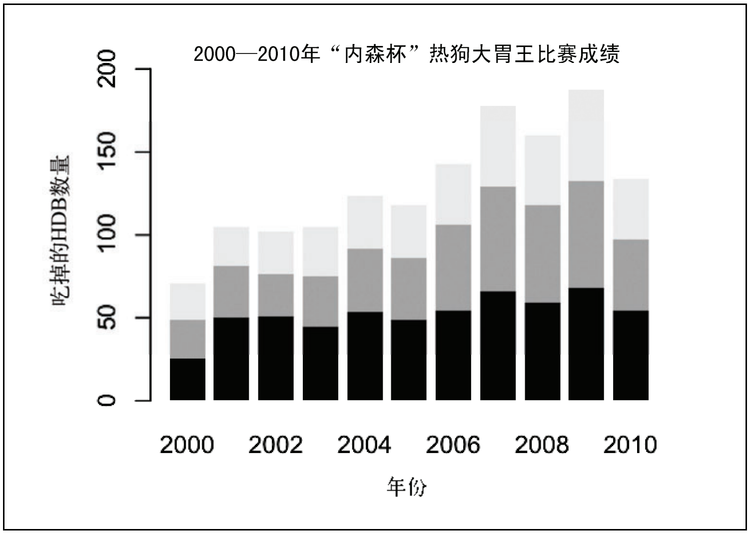


图4-22　利用R创建的堆叠柱形图

几行代码就能得到这个效果，很不赖。不过你还可以继续完善它。将图形存储为PDF格式，并用Illustrator打开。通过之前用过的那些工具，你可以添加文本、修改字体、简化垂直轴、利用“选择相同填充色的对象”功能来统一编辑颜色，当然别忘了还要标明数据来源（见图4-23）。

最后添加一段文字说明，再修改一下图表的总标题，图4-21显示了最终的结果。

我们在下一章会认识堆叠柱形图的表兄弟：堆叠面积图。它们在几何上非常相似，但堆叠面积图可以理解为把所有的堆叠都连通变成一段持续的曲线。

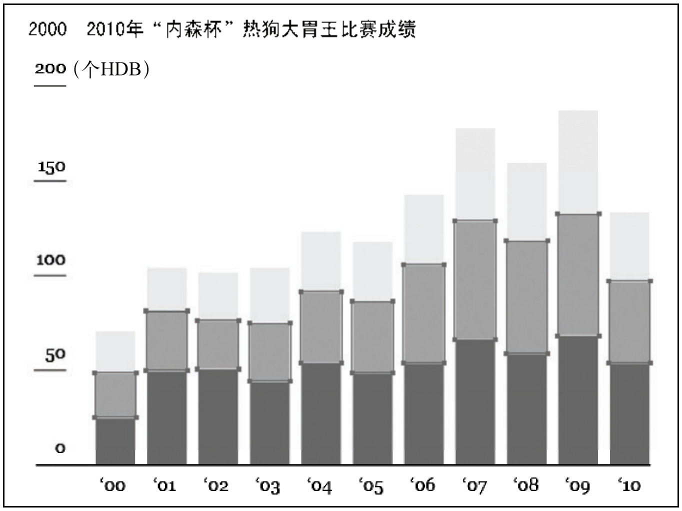


图4-23　在Illustrator中编辑

4.2.3　圆点

有时候，如果用圆点代替柱形，表达的意思可能会更加明确。它们占据的空间小，而且不是矩形，所以能更好地体现出“流”的感觉。图4-24显示了用点状图表来表现时间数据时的几何特征。

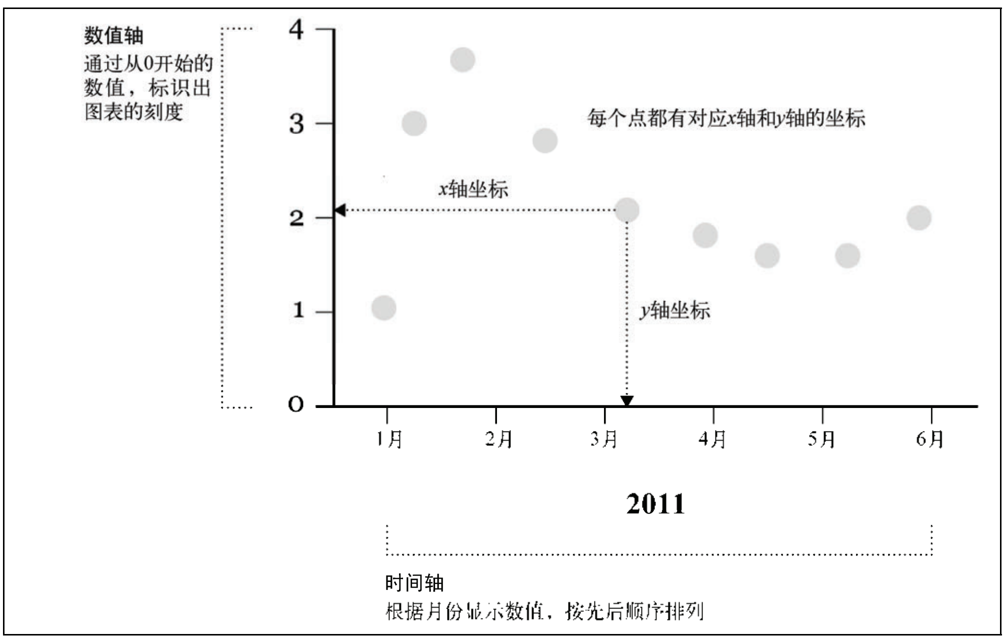


图4-24　点状图表的基本框架

这种类型的图表通常被称为散点图（scatterplot），你也可以用它来表现非时间数据。它通常用于表现两个变量之间的关系，我们会在第6章中谈到相关内容。而对于时间数据，我们通过水平轴来表示时间，数值或量度则在垂直轴上表现。

柱形图用高度作为视觉线索，而散点图则用位置。你可以根据每个点的*x*轴和*y*轴坐标来观察，并且根据其他点的位置来进行相互比较。出于这一原因，散点图的数值轴不必总从0开始（不过从0开始通常是不错的）。

创建散点图

利用R中的`plot()`函数可以很容易地创建散点图，不过也可以根据不同的数据类型稍加变化。图4-25显示了最终的图表。

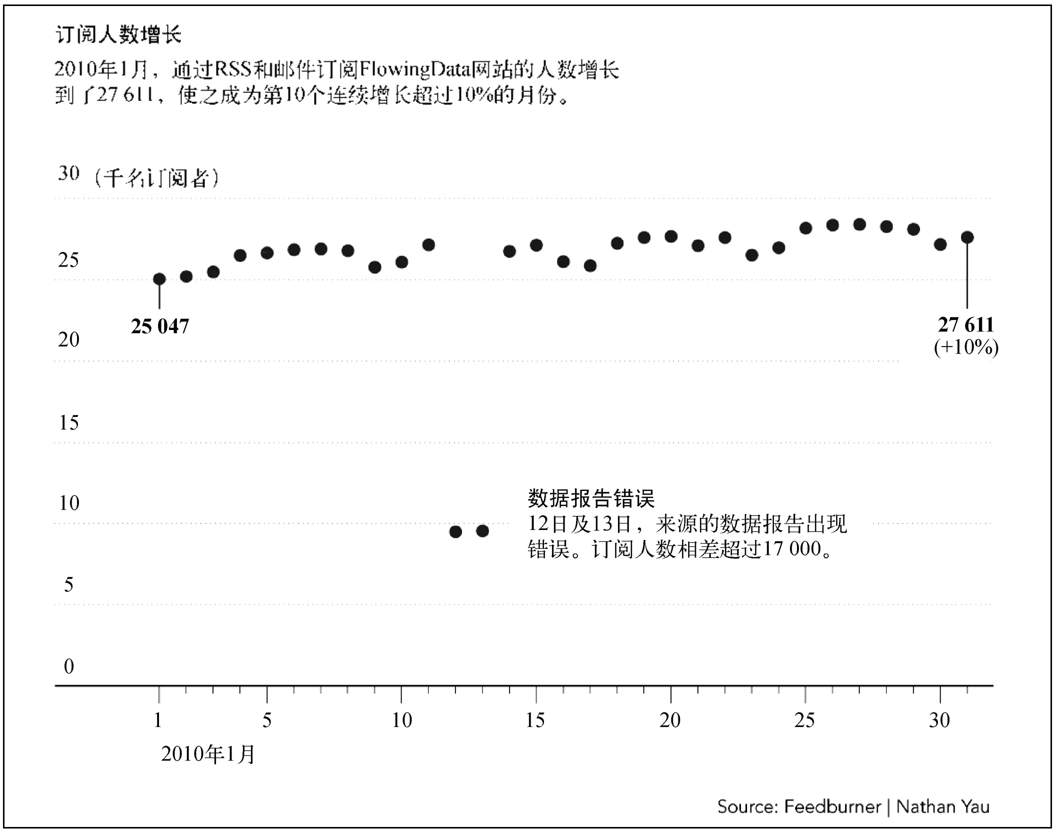


图4-25 在R中创建、在Illustrator中设计得到的散点图

这次的故事是关于FlowingData网站于2010年1月计算的订阅者数量，数据报告来源于Feedburner，该服务能够追踪每天有多少人阅读FlowingData上的文章。在2010年1月1日，订阅人数是25 047，而到月底的人数是27 611。不过其中最有意思的地方可能是该月中旬的明显下滑。我是否说了什么话冒犯了17 000名读者，让他们愤然退订？应该不是这么回事。

提示　数据并不都是真实的。可能是笔误、报告错误或其他原因导致数据偏离了现实。

还记得在R中创建图表时要做的第一件事是什么吗？载入数据。用`read.csv()`直接从URL中获得数据。

```
subscribers <-
    read.csv("http://datasets.flowingdata.com/flowingdata_subscribers.csv",
    sep=",", header=TRUE)
```

输入以下代码可以查看数据的前五行：

```
subscribers[1:5,]
```

看上去应该是这样：

```
        Date Subscribers Reach Item.Views  Hits
1 01-01-2010       25047  4627       9682 27225
2 01-02-2010       25204  1676       5434 28042
3 01-03-2010       25491  1485       6318 29824
4 01-04-2010       26503  6290      17238 48911
5 01-04-2010       26654  6544      16224 45521
```

可以看到有5列数据，分别对应日期、订阅人数、访问网站人数、浏览文章数和网站点击数。我们只关心其中的订阅人数。

你也可以让R把日期也绘制出来，但由于每天的人数本来就是按时间顺序排列的，所以第一列的内容其实并无太大必要。输入下面这行代码就能看到点状图，效果如图4-26所示。

```
plot(subscribers$Subscribers)
```

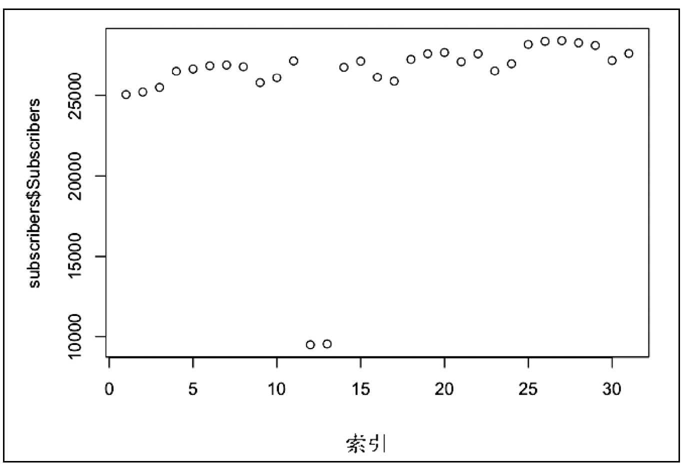


图4-26　R中的默认图表

很简单，对吧？`plot()`函数可以让你创建多种不同类型的图表，不过点状图是它的默认类型。你只用到了数据中的订阅人数。当你只给`plot()`函数提供一个数组时，它会假设该数组中

含有数值，并且自动为*x*轴坐标建立索引。

现在明确指定你希望的是点状类型，并且设置垂直轴的范围是0 ~ 30 000。

```
plot(subscribers$Subscribers, type="p", ylim=c(0, 30000))
```

图4-27和图4-26是一回事，但垂直轴的范围更宽，因为你用`ylim`参数进行了指定。请注意你用`type`参数告诉了R使用点状类型（`p`）。如果你把类型改为`h`，R就会改为创建高密度的垂直线。

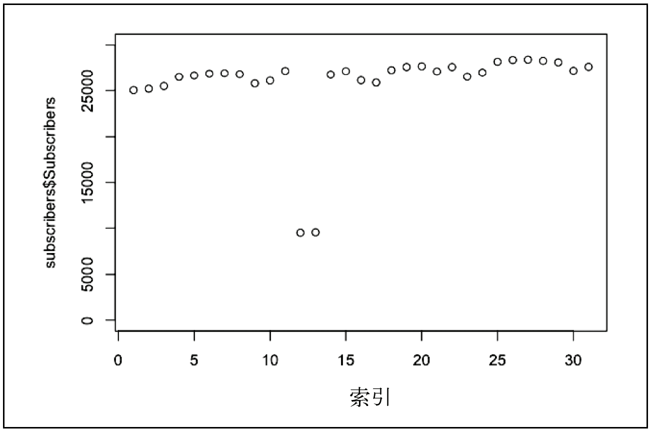


图4-27　R中指定了*y*轴范围的点状图

4

你也可以合并这两种类型，如图4-28所示。不过你还需要用到`points()`函数。这是因为只要一用到`plot()`函数，你就会创建一个新的图表，而不是在已有图表上添加新元素。如果你打算合并垂直线和点状图这两种效果，以下是需要的代码。

```
plot(subscribers$Subscribers, type="h", ylim=c(0, 30000),
    xlab="Day", ylab="Subscribers")
points(subscribers$Subscribers, pch=19, col="black")
```

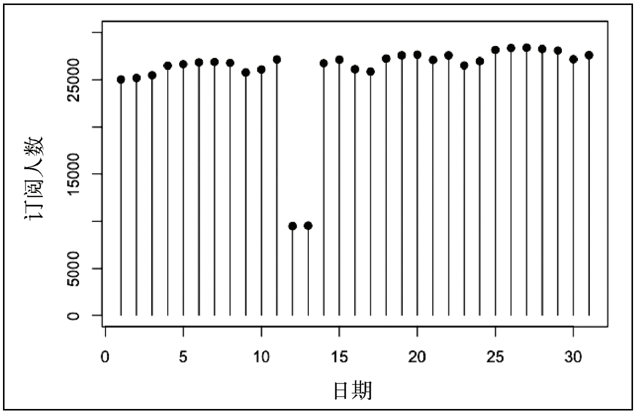


图4-28　带有高密度垂直线的点状图

垂直线类型的图表会首先绘制出来，这一次加上了坐标轴标签。之后点状图会添加到已有的图表中。pch参数表示用什么字符来标注点，而col参数我们曾在柱形图中用到过，它指定填充颜色。

现在让我们回到图4-27。将其保存为PDF文件，然后在Illustrator中打开，以便进行一些设计工作。

用选择工具选中标签，改成你喜欢的字体。然后打散标签群组，以便能单个编辑垂直轴的标签。通过“Transform→Transform Each”将标签正过来，然后用直接选择工具删除y轴上的垂直线。你不一定非要这么做，只不过它有点占地方。

最后选中那些圆点，它们目前是白色的圆圈。调整“颜色”面板中的选项，设置你想要的填充色和边框色。可能你还需要打开“颜色”面板的选项菜单（见图4-29），把颜色模式从灰度（Grayscale）改成CMYK（也就是青、品红、黄和黑四色搭配模式），这样才能有更多的用色选择。



图4-29　“颜色”面板中的选项菜单

这样你就会得到图4-30。它已经很像最终的图表了。

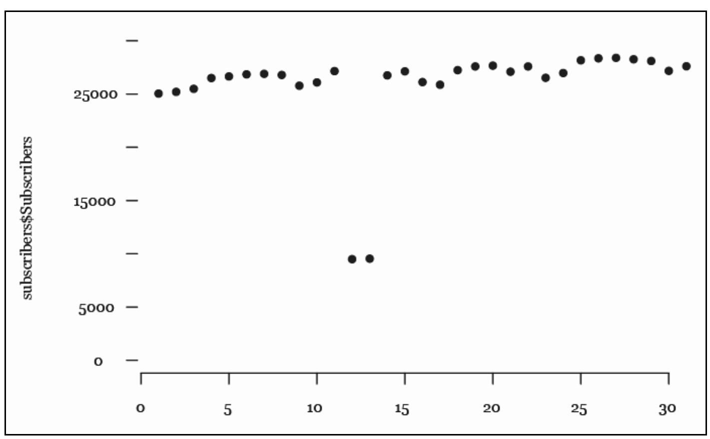


图4-30　调整坐标轴和颜色之后的点状图

现在试着添加网格，这样一来读者就能更容易观察右边较远圆点所代表的值，以及和之前各圆点之间的关系。用选择工具选中数值轴上的刻度线，然后拖动鼠标将它们拉长到贯穿整个图表。现在看来它们有点生硬，所以需要改变线条的样式。和之前一样，你可以在"描边"面板中做到这一点。如果你想用较细的虚线，可以按图4-31那样设置。

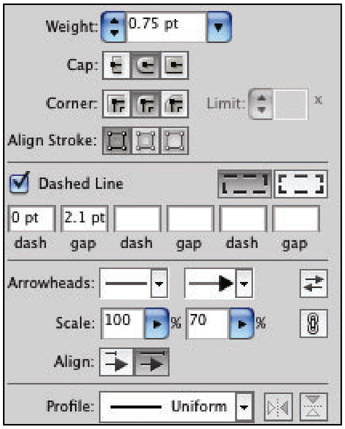


图4-31　"描边"面板中虚线样式的各种选项

图4-32是修改后的效果，同时添加了数值的单位。

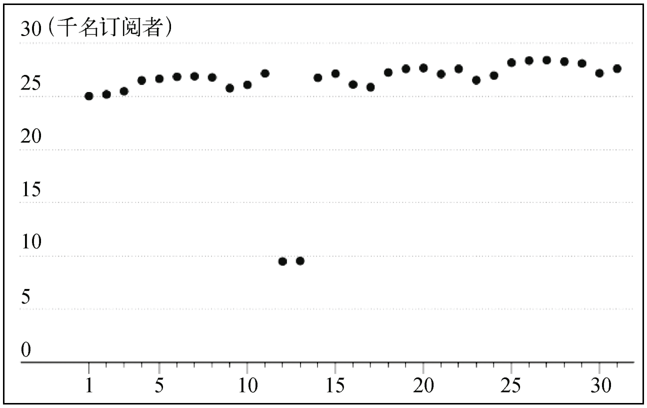


图4-32　添加网格并且在数值轴上标明数值和单位

在图4-32的基础上，使用同样的工具和技巧以实现最终的图表。用钢笔工具在水平轴上绘制刻度线，用文字工具编辑和添加标签。不要忘了在图表底部加上数据来源，让故事更加完整。

4.3　延续性数据

延续性时间序列数据的可视化和离散型数据的可视化很相似。因为就算是延续性数据，你手

中掌握的数据集依然还是离散且有限的。延续型和离散型在结构上并无二致，区别在于它们所呈现的真实世界。正如之前所提到的，延续性数据表现的是不断变化的现象，而为了实现这一目的，我们应当选择合适的可视化形式。

4.3.1　点与点相连

你可能对这种图表也比较熟悉。时间序列图表的绘制和点状图很相似，只不过你还要用线条将圆点连接起来。这些圆点一般也不用显示。图4-33显示了这一颇受欢迎的图表类型的几何特征。

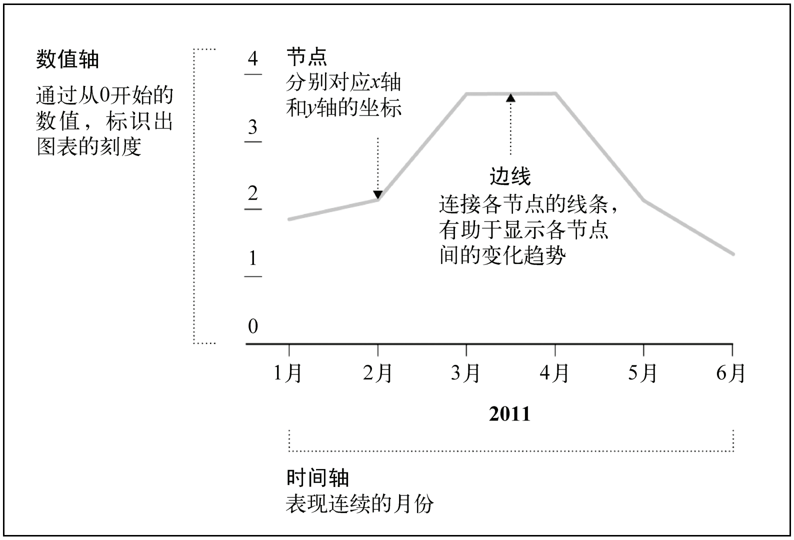


图4-33　时间序列图表的基本框架

图中有节点（或者说圆点，带有各自的*x*轴及*y*轴坐标）和边线（或者说连接线，以便观察数据间的变化趋势）。数值轴从0开始比较好，因为从其他数值开始可能会影响到图表的比例范围。

水平轴的长度也会影响到观察到的趋势。如果水平轴过短，点与点间的增长就会看起来比较夸张。如果水平轴过长，你可能就无法发现其中的变化模式。

创建时间序列图表

如果你学会了如何用R创建散点图，那么就能掌握创建时间序列图表的方法。载入数据，然后使用`plot()`函数，但在`type`参数中不用`p`，而是用`l`（代表线条，line）。

我们这次用世界银行发布的1960—2009年间的世界人口数据来作演示。和往常一样，用`read.csv()`函数来载入数据。

```
population <-
    read.csv("http://datasets.flowingdata.com/world-population.csv",
    sep=",", header=TRUE)
```

以下是最上面几行数据的样子。只有年份和人口数量两种数据。

```
  Year Population
1 1960 3028654024
2 1961 3068356747
3 1962 3121963107
4 1963 3187471383
5 1964 3253112403
```

用`plot()`函数并指定*x*轴和*y*轴坐标、图表类型、数值轴范围和坐标轴标签。

```
plot(population$Year, population$Population, type="l",
ylim=c(0, 7000000000), xlab="Year", ylab="Population")
```

你的图表应该如图4-34所示。

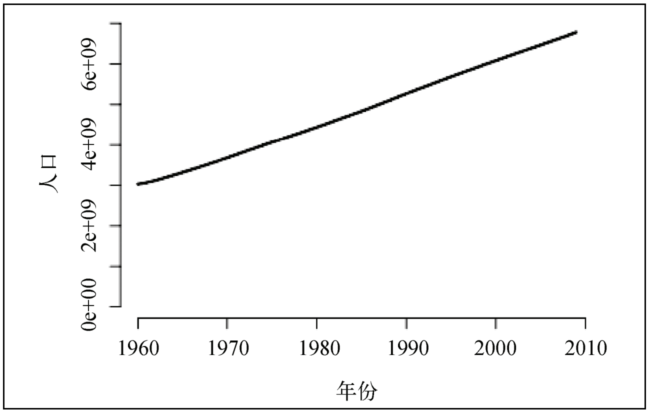


图4-34　R中的默认时间序列图表①

现在你可以将图形保存为PDF格式，然后在Illustrator中编辑它。这个操作我们已经做过好几次了，现在尝试一点新花样：用Illustrator的折线图（Line Graph）工具来设计整张图表。Illustrator中提供了数个图表工具，可以让设计师直接创建一些基本图表，折线图工具就是其中之一（参见图4-35）。



图4-35　Illustrator中的图表工具

首先在“工具”面板中选择折线图工具。你会看到一些图表类型的图标。单击各图标选择图表类型。

然后从http://datasets.flowingdata.com/world-population.csv下载人口数据。你不能像在R中那样直接通过URL载入数据，因而必须把文件存在自己的电脑里。用Excel或者Google Documents打开这个CSV文件，如图4-36所示。复制所有的行（第一行除外，因为其内容是各列的名称）。现在

① 图中纵轴数值是用科学计数法表示的，举例来说，$2e + 09 = 2 \times 10^9$，其他值可类推。——编者注

要把数据粘贴到Illustrator里面去。

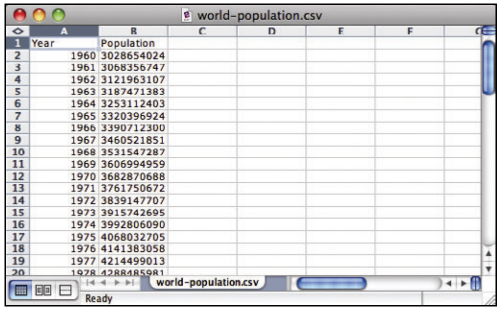


图4-36　在Excel中打开的CSV文件

回到Illustrator。使用从“工具”面板中选择的折线图工具，通过单击并拖动鼠标画一个矩形，其大小约等于你想要的图表尺寸。此时会弹出一个电子表格，如图4-37所示。

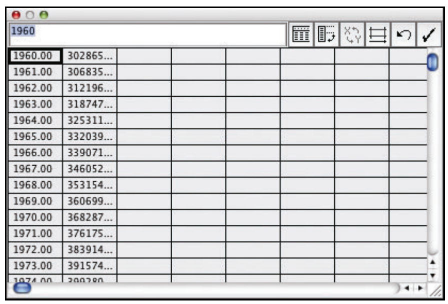


图4-37　Illustrator中输入数据的电子表格

把你刚才从Excel里复制的数据粘贴进来，然后单击右上角的勾号。你现在应该能看到类似图4-38的图形。

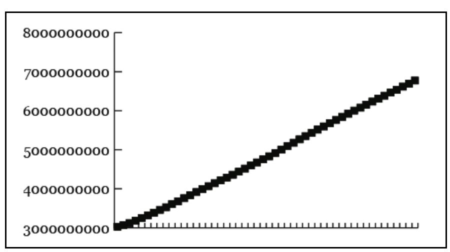


图4-38　Illustrator中默认的折线图

已经基本成型了，不过我们还需要调整一些选项，以让图表看上去不那么粗糙。鼠标右击图表并选择“类型”（Type）。如图4-39中那样，取消勾选Mark Data Points（标记数据点）前面的复选框。

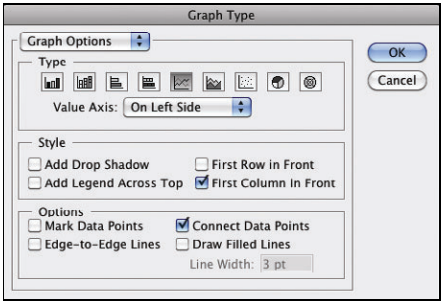


图4-39 Illustrator中的图表选项

在下拉菜单中选择“类别轴”（Category Axis）选项，然后将刻度线长度设置为“无”（None），单击确定按钮。这会让你的图表看起来更加干净而不那么杂乱。此后的设计步骤和之前由R生成的图表是一样的。

提示 如果你打算绘制一些基础的图表类型，而且要用Illustrator来编辑，那么可以直接在Illustrator里面创建它，这样可以节省时间。并不是所有图表都得先在R里面创建。当然了，也不是所有东西都得在Illustrator里面创建。

你可以清理一下垂直轴、简化数值标签、在水平轴上添加刻度线和年份标签，并且添加标题和说明文字。你还可以调整曲线的样式，让它更加突出。默认的浅灰色曲线给人感觉像是背景元素，而我们应该让它突出显示在画面中央。完成这些调整之后，你的图表效果应该类似图4-40那样。

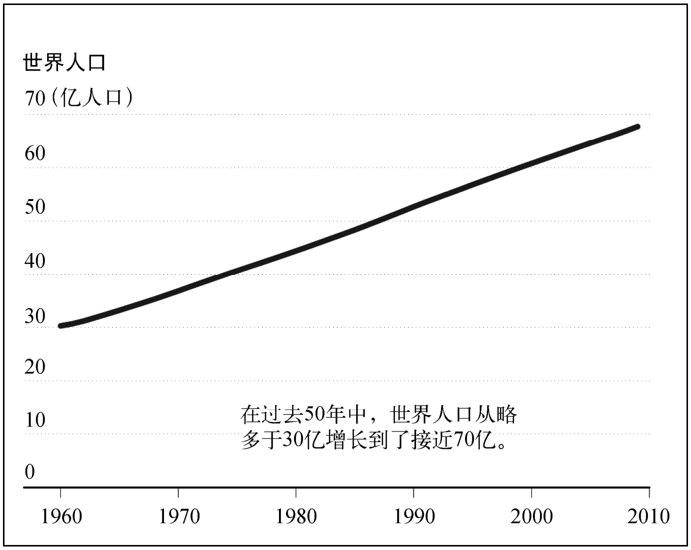


图4-40 过去50年中的世界人口

这一节想表达的意思是，你可以用Illustrator和R创建出同样的图表——二者生成的结果是相同的。你习惯使用哪种工具就用哪种工具好了，最重要的是得到结果。

4.3.2　一步一个台阶

标准折线图表的缺陷之一是它必须要表现从*A*点到*B*点间的稳定变化。它对于表现世界人口这样的数据是没问题的，但有些事物会长时期停留在某个值上，然后突然出现增长或者衰退。比如说，银行利率就会保持几个月不变，然后某天突然下调。对于这种类型的数据可以用阶梯图，如图4-41所示。

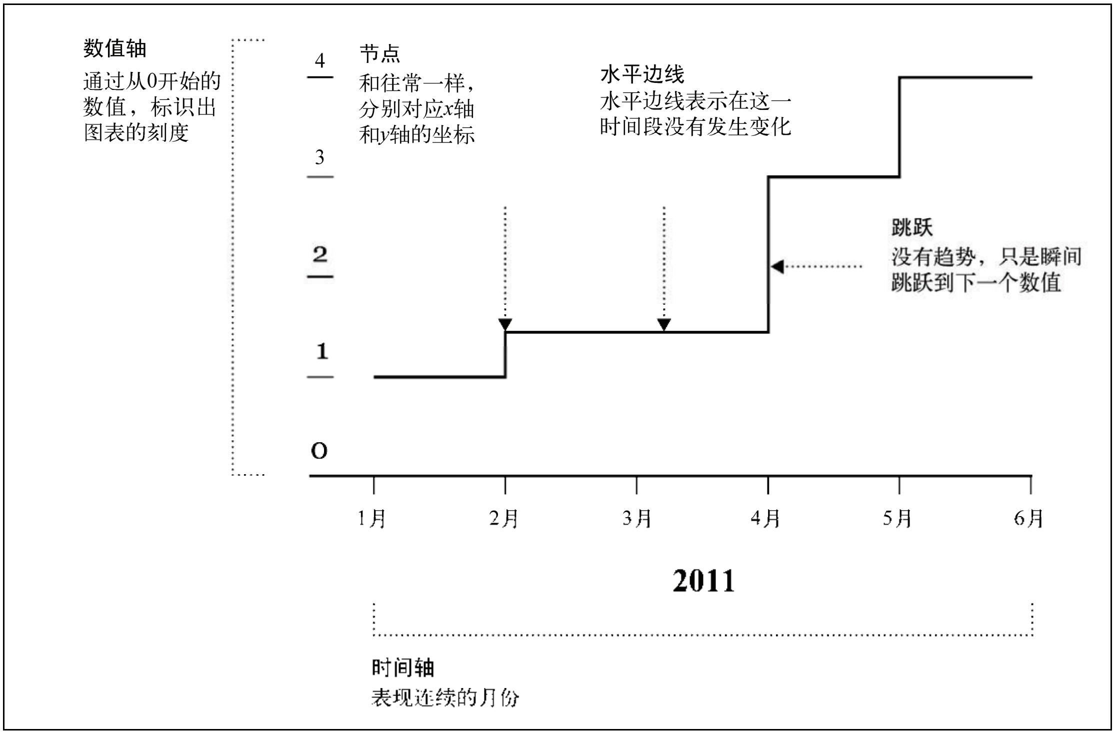


图4-41　阶梯图的基本框架

与直接连接*A*点和*B*点不同，曲线会保持在同一数值，直到发生变化，那时再直接向上（或向下）跳跃到下一个数值。一般会有多个这种阶梯。

创建阶梯图

Illustrator里面没有能直接创建阶梯图的工具，但是R里面有，所以你可以先在R里面创建基础图表，然后在Illustrator里面进行编辑。你是否已经发觉了这其中的模式？

图4-42显示了最终的图表。它表现的是美国邮政投递信件的资费变化。请注意，这些变化并不是定期发生的。从1995—1999年，邮费一直保持在32美分，4年间并无涨落。然而在最近的

2006—2009年间，每年都出现了上涨。

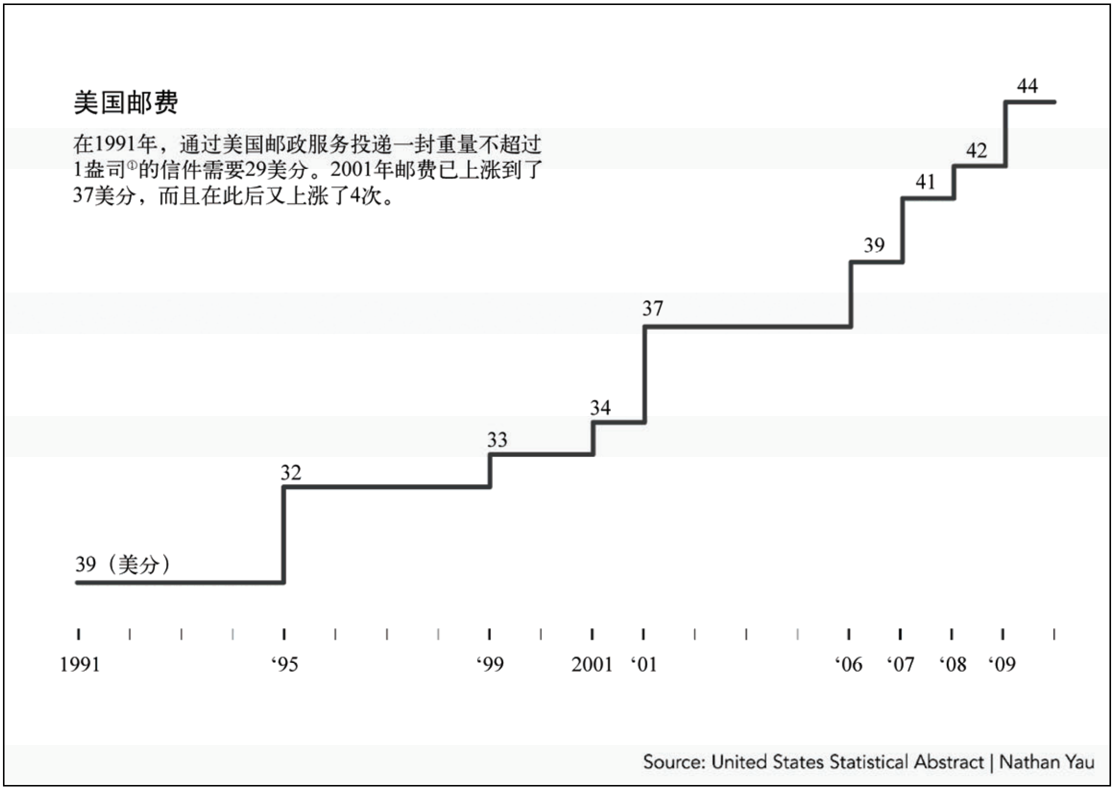


图4-42 显示邮费变化的阶梯图

要在R中创建阶梯图，同样需要经过本章中你一直遵循的几个步骤：

(1) 载入数据；

(2) 确保数据有适当的格式；

(3) 通过某个R函数来创建图形。

你可以在美国统计摘要的网站上找到邮政资费的历史数据。我已经把它们整理为一个CSV文件，地址是http://datasets.flowingdata.com/us-postage.csv。直接将URL插入到`read.csv()`中，让它作为R的数据源。

```
postage <- read.csv("http://datasets.flowingdata.com/us-postage.csv",
 sep=",", header=TRUE)
```

以下列出了完整的数据集，其中只有10个数据节点，分别代表1991—2009年间的每一次邮费变动，最后一个数据节点表示当前资费。第一列是变化发生的年份，第二列是资费，单位为

① 1盎司 = 28.350克。——编者注

美元。

```
   Year Price
1  1991  0.29
2  1995  0.32
3  1999  0.33
4  2001  0.34
5  2002  0.37
6  2006  0.39
7  2007  0.41
8  2008  0.42
9  2009  0.44
10 2010  0.44
```

用`plot()`函数来创建阶梯图很容易。将年份作为*x*轴坐标、价格作为*y*轴坐标，然后将类型设置为`s`，自然它代表的是阶梯（step）。

```
plot(postage$Year, postage$Price, type="s")
```

如果你愿意，还可以指定标题和坐标轴标签。

```
plot(postage$Year, postage$Price, type="s",
    main="US Postage Rates for Letters, First Ounce, 1991-2010",
    xlab="Year", ylab="Postage Rate (Dollars)")
```

这样就得到了邮费的阶梯图，如图4-43所示。

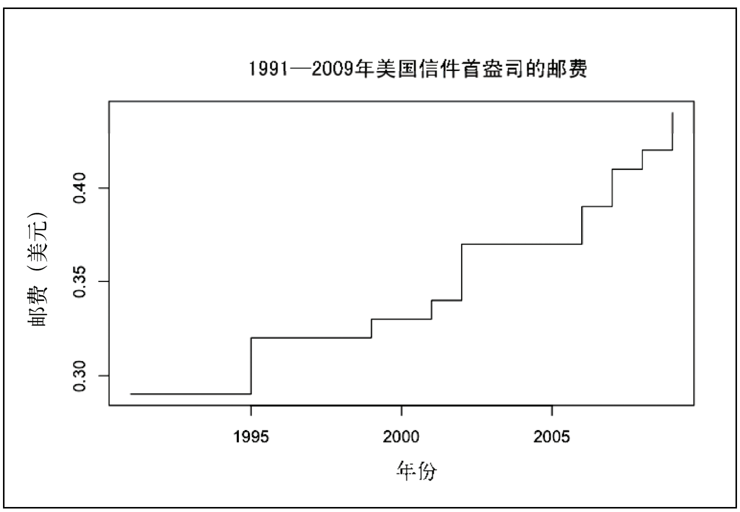


图4-43　利用R创建的阶梯图

有兴趣的话，你还可以试试看折线图的效果（参见图4-44）。

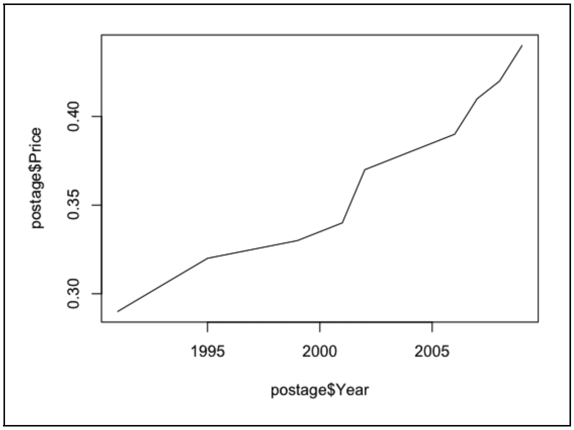


图4-44 邮费的折线图

图中体现出了增长的趋势，但你是否注意到它看上去好像是在稳定增长？虽然在2001—2006年间并没有38美分的资费，但从折线图中你无法得知这一点，除非看到原始数据。

将阶梯图存储为PDF格式，然后在Illustrator中打开。重复之前的过程，按你的喜好来编辑图表。我去掉了整个垂直轴，直接在每一个跳跃处添加标记（使用输入文字工具）。同时我在水平轴上绘制了平均分隔的刻度线，但只在发生变化的年份添加标记。

提示 如果数据集比较小，可以直接在数据节点上进行标记，而无需使用数值轴和网格。这样会更加强调数据本身。而且由于数据节点并不多，因而标记并不会显得过于杂乱。

最后，我使用了灰色背景。这固然是出于个人喜好，但这种背景色有助于强调图表，尤其是当图表隐没在文本中时。它提供了一个包容的空间，同时又不会显得刺眼。要想在所有图形和文字之后创建背景，你需要在Illustrator中创建一个新的图层。可以在“图层”（Layers）面板中进行设置。单击面板右下角倒数第二个按钮来创建新图层。新建的图层默认会被放置在最上方，但我们希望它退后到最底层，所以需要把新图层拖到Layer 1图层的下方。然后你还可以对层进行重命名。这个操作在你开始设计更复杂的图形时会为你带来极大的帮助。我将新图层重命名为“background”，如图4-45所示。

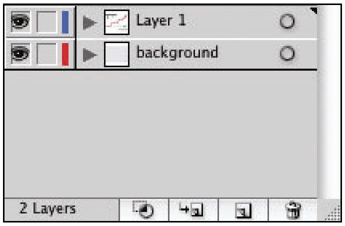


图4-45 Illustrator中的“图层”面板

在此之后，用矩形工具（Rectangle tool）绘制一个矩形，并且按你的想法来改变它的大小，然后在“颜色”面板中改变它的颜色。

4.3.3　平滑和估算

如果你手中的数据太多，或者数据杂乱无章，可能就会很难辨认其中的趋势和模式。为了改善这一点，你可以估算出一条趋势线。图4-46显示了基本的想法。

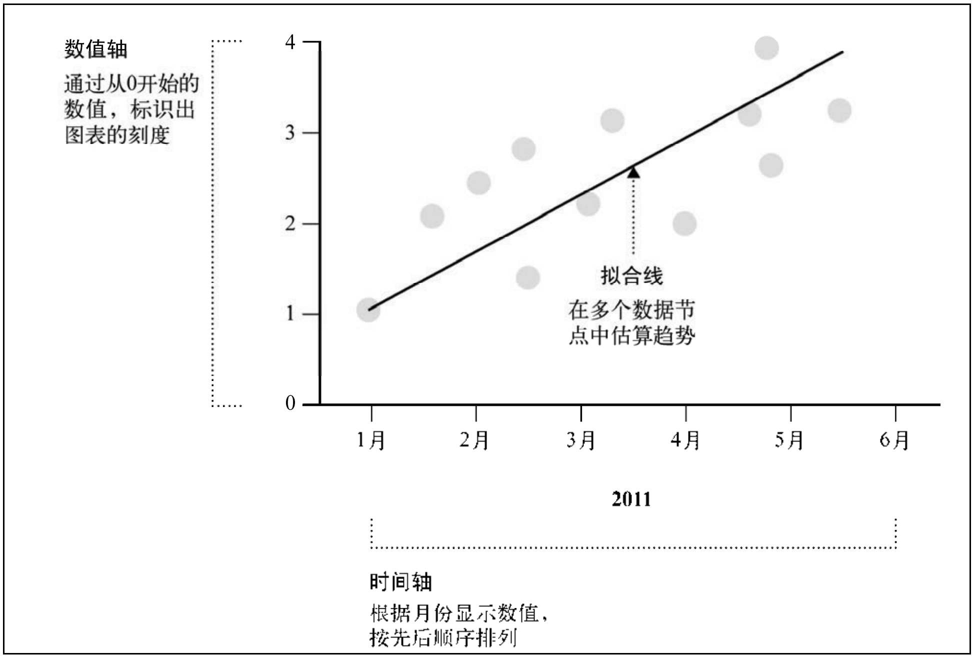


图4-46　为数据节点拟合一条线

绘制一条线穿过尽可能多的数据节点，同时尽量减少各数据节点与拟合线之间的总距离。最直截了当的做法就是创建一条直线，这其中用到了你在中学时学过的基础斜截式方程：

$$y = mx + b$$

其中m代表斜率，b代表截距。但如果你的趋势不是线性的该怎么办？用直线是不可能拟合带有波峰波谷的数据的。在这里我们可以用到William Cleveland和Susan Devlin创建的一种统计学方法，它被称作LOESS，即局部加权散点平滑法（Locally Weighted Scatterplot Smoothing）。通过它你能用曲线来拟合数据。

LOESS从最开头的数据开始对数据进行“切片”。针对每一个切片中的数据，它都会估算出一个低阶多项式。LOESS会分析每一段数据，拟合一大堆微小的曲线，然后将它们合并形成一条曲线。你可以Google一下以了解更多细节，有关这一主题有不少论文。现在让我们来看看怎样用LOESS处理数据。

> ▶为了解LOESS的完整细节，参阅William Cleveland在《美国统计协会杂志》上发表的文章"Robust Locally Weighted Regression and Smoothing Scatterplots"（强劲的局部加权回归及平滑散点图）。

拟合一条LOESS曲线

你现在看到的故事是美国过去几十年间的失业问题。失业率一直有升有降，而且呈现出季节性波动。但其中的整体趋势是怎样的？如图4-47所示，美国失业率在20世纪80年代达到了顶峰，90年代一直在下降，然后在2008年前后开始反弹。

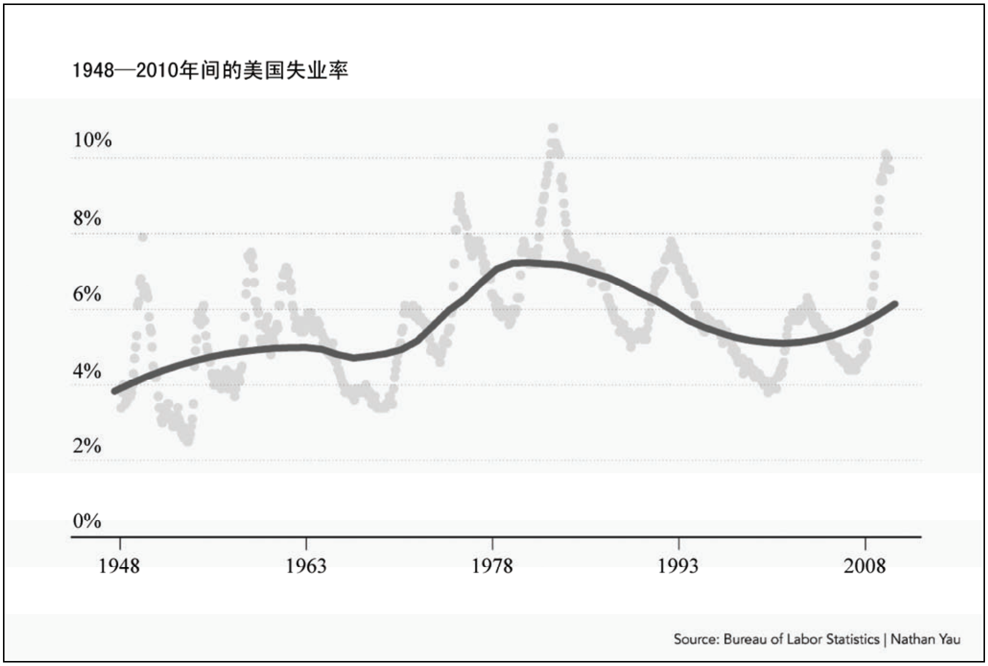


图4-47 用LOESS曲线拟合的美国失业率

图4-48显示了用R的`plot()`函数生成的失业率点状图表。

```
# 载入数据
unemployment <-
    read.csv(
        "http://datasets.flowingdata.com/unemployment-rate-1948-2010.csv",
        sep=",")
unemployment[1:10,]

# 生成散点图
plot(1:length(unemployment$Value), unemployment$Value)
```

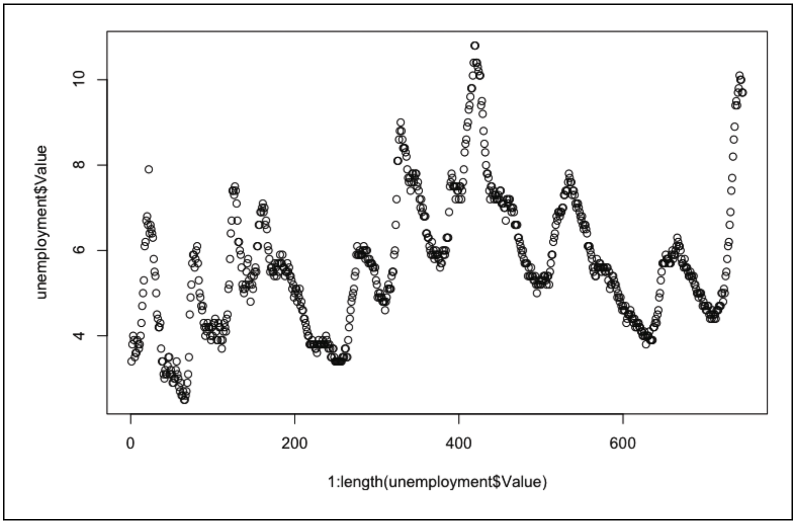


图4-48　只有节点的失业率图表

图4-49显示了直线拟合的效果。

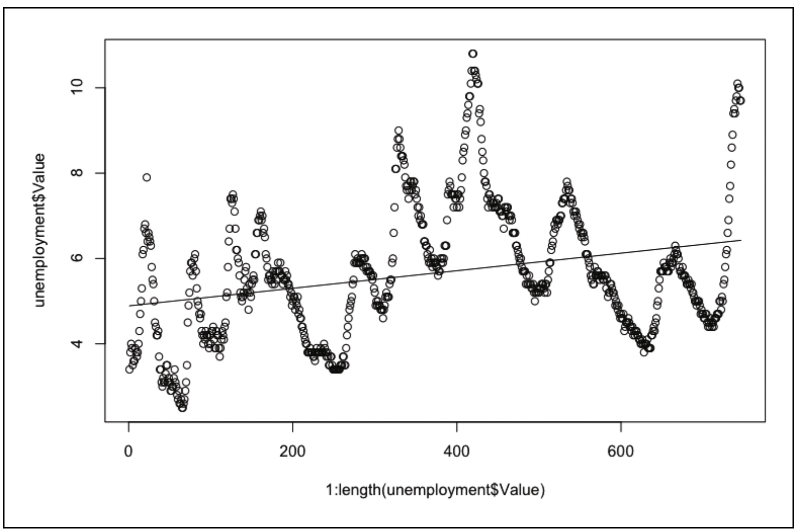


图4-49　笔直的拟合线

这并没有什么用。它看起来好像忽略了失业率的所有起伏波动。要想用LOESS曲线来拟合，我们得用到`scatter.smooth()`函数。

```
scatter.smooth(x=1:length(unemployment$Value), y=unemployment$Value)
```

运行结果如图4-50所示。线条开始向上凸起，对应了20世纪80年代的失业高峰。现在有点感

觉了。

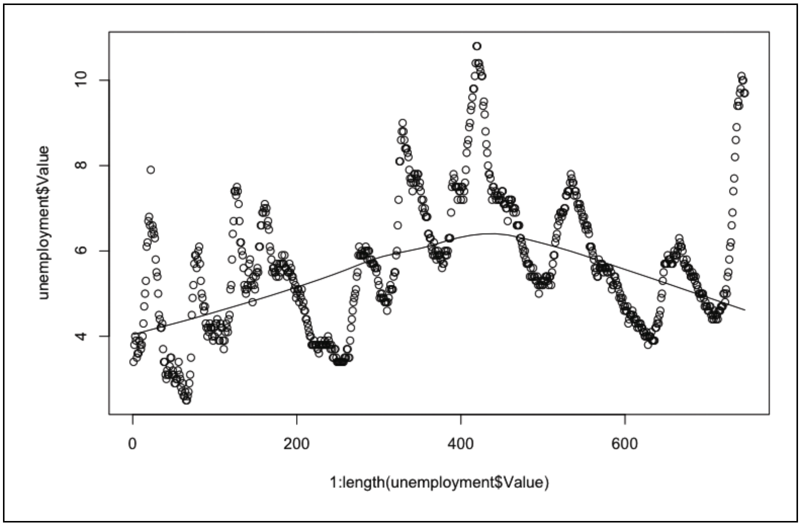


图4-50　LOESS曲线的拟合

你可以通过scatter.smooth()函数中的degree和span参数来调整曲线的拟合度。前者控制拟合多项式的阶数，后者控制曲线的平滑度。span越接近于0，拟合的程度就越高。图4-51显示的是degree为2、span为0.5时的效果。现在我们再调整一下颜色和坐标轴的范围。

```
scatter.smooth(x=1:length(unemployment$Value),
    y=unemployment$Value, ylim=c(0,11), degree=2, col="#CCCCCC", span=0.5)
```

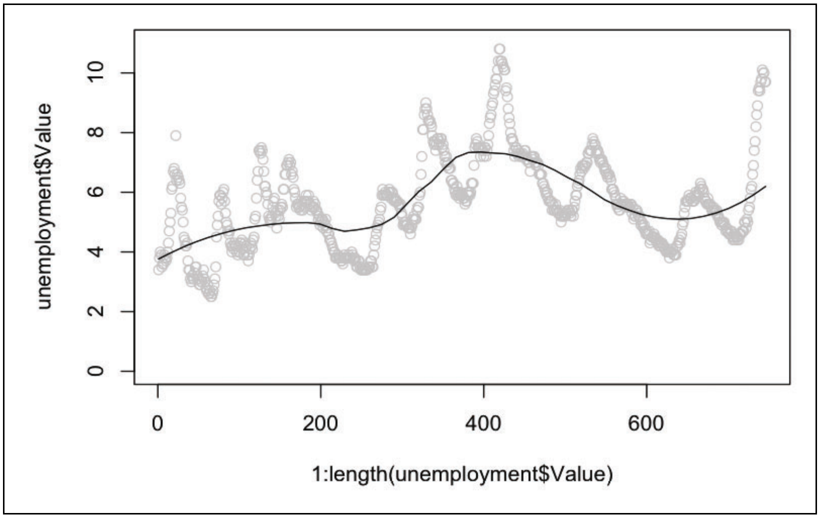


图4-51　平滑度较小、多项式阶数较高的拟合LOESS曲线

通过这些设置，曲线的上下起伏更加突出了。你可以多试试不同的span值，了解一下它是如何改变曲线平滑度的。

为了得到图4-47中的最终效果，将图形存储为PDF并在Illustrator中打开。使用同样的工具（比如选择工具、文字工具、钢笔工具）添加标题、背景色和水平轴的刻度线。拟合线比数据节点要突出得多，这样可以把读者的注意力引导到趋势上面。

提示　使用不同的颜色、笔触粗细来强调图表中的重要部分。

4.4　小结

探索有关时间的模式是非常有趣的。时间和我们的日常生活是如此不可分割，以至于时间数据的可视化中有很多方面都是相当直观的。我们能够理解事物正在变迁和发展——难的是弄清楚到底变化了多少，以及懂得要从图表中挖掘什么内容。

看一眼图表中的线条，我们可以很快了解到事物是在上升还是在下降。这就是可视化的目的——快速了解数据的概貌。但我们还能更进一步。你可以利用可视化作为探索的工具。仔细观察某个时间段，尝试解释为什么在这一天会有小波动，或者为什么在那一天又会突然上升。这才是数据有意思的地方——你对数据越了解，讲出来的故事也会越精彩。

在理解了数据的内容后，你需要在数据图表中对其中的细节作出解释。强调那些令人感兴趣的部分，引导读者去关注它们。过于简单朴素的图表对你自己而言没有什么问题，但由于没有上下文背景，在其他人看来就会是索然无味的。

为了实现这一目的，我们使用了R和Illustrator。R用于基础的搭建，而Illustrator用于图表的设计，指出数据中的重要部分。当然，本章中所涉及的图表类型只是其中的一部分，它们只和时间数据相关。而如果往图表中引入动画和交互，你又将掌握很多新鲜花样，大家将会在下一章中看到。同时你会了解一种新的图表类型——比例图。本章中所用到的编程步骤和设计原则同样适用于下一章，即使你用另一门语言来写代码。

有关比例的可视化

5

时间序列数据自然是以时间为分组依据的。它显示了在某个特定时间范围内发生的一系列事件。在比例数据中同样也存在分组依据，只不过它是按类别、子类别和群体进行划分的。这里的群体并不仅指人类群体，它代表的是各种可能的选择或产出，也就是样本空间（sample space）。

问卷调查可能会询问人们对某个问题是赞同、反对还是保持中立。每一个观点类别都有各自的意义，而且各部分加起来构成了整体。

本章讨论如何表现单个的类别，并提供了更广阔的视野——各个选择之间如何相互关联。你会用到之前章节中学到的工具，还会首次尝试用HTML、CSS和JavaScript创建可交互的图表，并接触到用Flash创建的图表。

5.1　在比例中寻求什么

对于比例，我们通常会寻求三件事情：最大值、最小值和总体分布。前两者很明确，将数据由小到大排列，位于两端的就是最大值和最小值。如果你处理的是投票结果，那么它们就代表参与者选择最多和最少的选项。如果你绘制的是食物各部分的卡路里含量，那么就会看到是哪个部分的卡路里最多，哪个部分的卡路里最少。

不过，表现最大值和最小值并不需要图表来呈现。我们真正最感兴趣的是比例的分配。投票中某个选项的结果和其他相比有什么不同？脂肪、蛋白质、碳水化合物都含有同样的卡路里吗？还是有某一种的卡路里含量占绝大多数？本章涉及的图表类型将会帮助你解答这些问题。

5.2　整体中的各个部分

这是形式最简单的比例呈现。你有一系列比例值，它们的总和为1，或者百分比的总和是100%。你既希望呈现各部分和其他部分的相对关系，同时又希望保持整体上的感觉。

5.2.1　饼图

饼图是最传统的图表之一。它们随处可见，比如商务演讲演示，甚至还有网站用它来开各种玩笑。已知最早的饼图是1801年由William Playfair发布的，他还发明了折线图和柱形图。真是有才。

你应该很清楚饼图的原理。如图5-1所示，我们从一个圆开始，用它来代表整体，然后把它们切成楔形，就和分蛋糕或披萨一样。每一个楔形都代表整体中的一部分。记住下面这句话，因为很多新手都会犯这个错误：所有楔形所占百分比的总和应该等于100%。如果不等，那就一定是哪里出了问题。

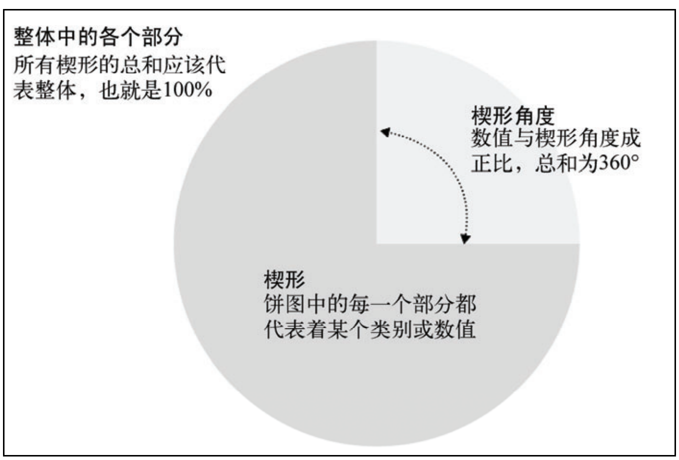


图5-1　饼图的基本框架

有些人对于饼图略持微词，因为它不像柱形图或者基于位置的图形那样精确，所以他们认为应该完全弃之不用。的确，衡量长度要比衡量面积或角度要容易得多，但这并不意味着我们就应该完全抛弃饼图。

只要了解了饼图的局限，使用起来并不存在问题。这一点很容易做到：良好组织数据，不要将一个饼图分成太多块。

创建饼图

几乎所有的图表工具都能绘制饼图，不过我们可以像上一章那样直接在Illustrator中创建。过程是一样的：首先添加数据，然后绘制基础图表，最后再加以修饰。

绘制基础图表（也就是饼图）的方法非常简单。创建一个新文件，从“工具”面板中选择Pie Graph Tool（饼图工具），如图5-2所示。按下左键并拖动鼠标创建一个矩形，其大小应该和你想要的图表尺寸相符。你还可以在后面调整尺寸。

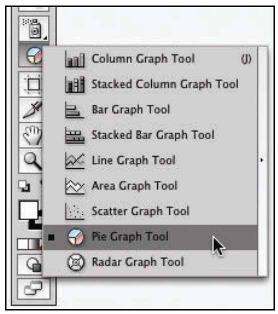


图5-2 Illustrator中的“工具”面板

在松开鼠标左键之后，会弹出一个电子表格窗口，要求你输入数据。如果是单个饼图，从左到右输入各个数据点，然后数据就会以同样的顺序显示在图表中。

我们这次以FlowingData网站发起的一项投票为例。网站列出了一系列与数据相关的领域，并要求访问者从中选择他们最感兴趣的选项。最后收到了831票。

感兴趣的领域	票　数
统计学	172
设计	136
商务	135
制图学	101
信息科学	80
网站分析	68
编程	50
工程学	29
数学	19
其他	41

在Illustrator的电子表格中输入数据，如图5-3所示。你输入数字的顺序会直接影响到饼图中各个楔形的顺序，它们从顶部开始，然后以顺时针方向依次排列。

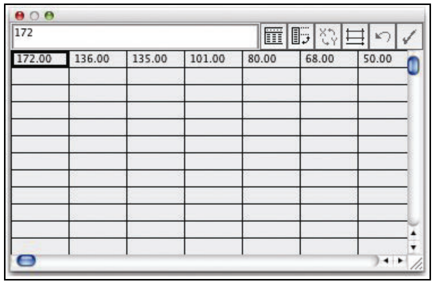


图5-3　Illustrator中的电子表格

请注意投票结果由大到小进行了排序，并且以“其他”类别作为结束。这种排序可以让你的饼图更加容易阅读。单击弹出窗口右上角的勾号，图表就创建成功了。

提示　Illustrator中的电子表格只是一个简要工具，你无法通过它对数据进行修改或者排序。解决办法是先用Microsoft Excel把所有的数据都处理好，然后再复制粘贴到Illustrator中去。

默认的饼图中出现了8个不同灰度的楔形，颜色似乎并无明确顺序，并且带有黑色边框，如图5-4所示。它看上去有点像一个灰色的棒棒糖，不过很容易就能改变这一点。最重要的是你已经得到饼图的雏形了。

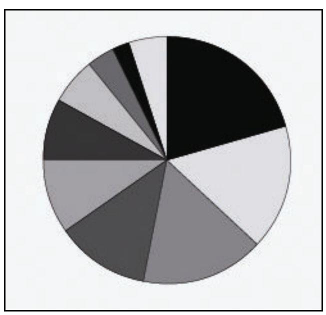


图5-4　默认的饼图

现在让我们来增强饼图的可读性，比如改变颜色、添加文本，以向读者解释他们看到的是什么。现在的饼图颜色并没有什么含义，它们的作用只是分开楔形而已，但我们可以利用颜色来告诉读者哪些是重点、遵循了何种顺序。毕竟你好不容易才把数据按大小排好序。

如果你从12点钟方向开始然后顺时针旋转，应该会发现数值的大小呈逐步下降的趋势。然而

现在的颜色是任意分配的，有一些较小的楔形反而颜色更深，得到了强调。颜色的深浅代表重点的强弱，因此我们应该让较大的楔形使用更深的颜色，而较小的楔形使用较浅的颜色。如果你希望强调那些获得票数较少的答案，那么就应该将颜色的深浅反过来。不过在这次投票中，你希望知道的是有哪些数据领域最受欢迎。

提示　颜色会极大地影响人们对图表的阅读。它不仅是美学元素——尽管有时候确实如此。颜色和长度、面积一样可以作为视觉线索，因此务必明智地进行选择。

在“工具”面板单击直接选择工具，然后选中某个楔形。通过“颜色”面板中的控件来改变填充色和边框色。图5-5显示了同样的饼图，但是边框变成了白色，而且各楔形按从深到浅排列。现在更容易识别出数值按从大到小排列了，不过最后一个楔形“Other”除外。

当然，在颜色上不必如此节俭。你可以使用任何喜欢的颜色，就像图5-6中那样。一般来说，不要使用过于明亮的颜色，因为你不想刺激到读者的眼睛。不过如果你的主题适合用晃眼的颜色，不妨放纵一下。

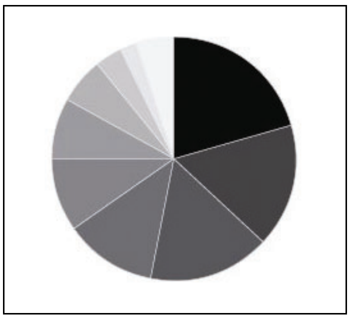


图5-5　颜色按从深到浅排列的饼图

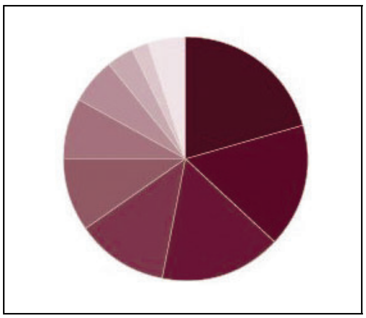


图5-6　着色后的饼图（另见彩插图5-6）

由于这是FlowingData发起的投票，我借用了FlowingData网站logo的红色系，然后以逐步降低不透明度的方法来逐渐减弱色彩。你可以在“透明度”（Transparency）面板中找到这个选项。不透明度为0时，填充色就会完全看不见；不透明度为100%时，填充色就不会有任何透明感。

说明　当你调整不透明度时，颜色会与背景色相混合。本例中的背景是白色的，所以透明度越高时颜色就会体现出褪色的感觉。但如果背景是蓝色的，颜色就会越来越紫。

最后，通过文字工具添加标题、文字说明以及各个标签。你可以任意选择自己喜欢的字体，不过在考虑文本的位置时，Illustrator的对齐工具会非常有用。将标签文字进行合理地、均匀地分布，这样能让你的图表更加易读。你还可以使用钢笔工具为最后三个投票类别绘制箭头（见

图5-7）。这些楔形的面积过小，不能把标签放到它们的内部，而且彼此间过于接近，标签放在邻近的位置可能会造成彼此混淆。

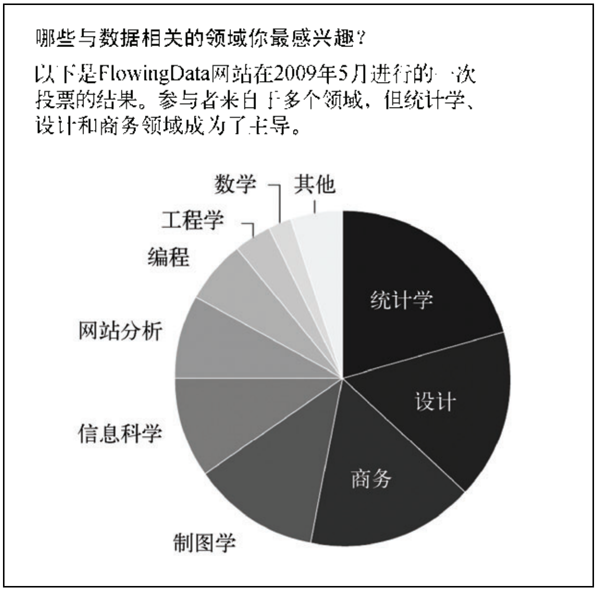


图5-7　带有标签和说明文字的最终饼图（另见彩插图5-7）

5.2.2　面包圈图

我们的好朋友饼图还有个小兄弟：面包圈图。它和饼图很相似，但在中间有一个洞，看起来更像面包圈，如图5-8所示。

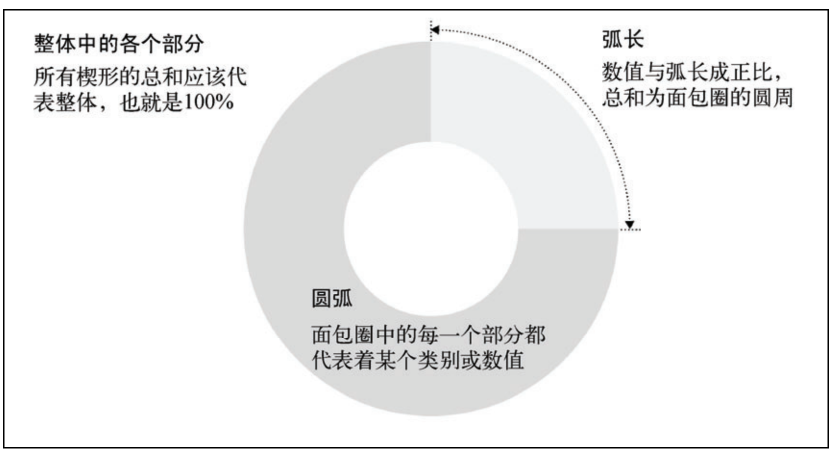


图5-8　面包圈图的基本框架

由于圆形中间有一个洞，你不能再通过角度来衡量数值了，而要通过各弧形的长度。如果在一个图表中包含过多类别，对这种类型的图表也会造成同样的问题，但如果类别较少的话，面包圈图用起来也是很顺手的。

创建面包圈图

在Illustrator里面创建面包圈图非常简单。像之前那样创建一个饼图，然后在中心处放置一个实心圆，如图5-9所示。同样用颜色来引导读者的视线。

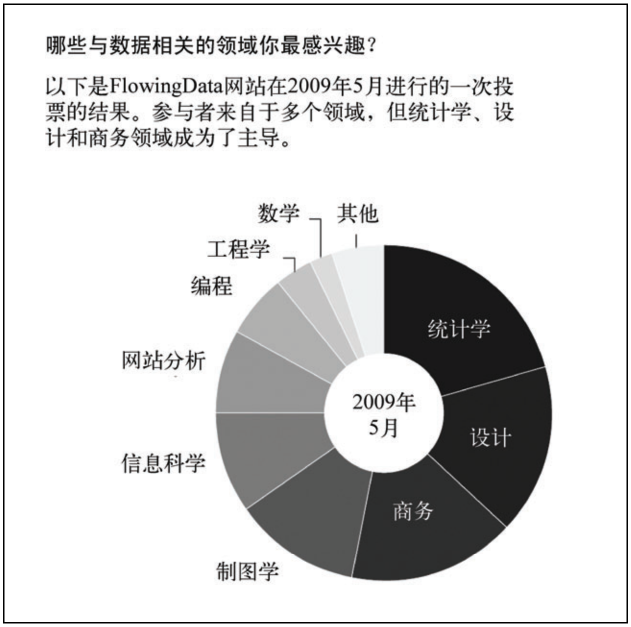


图5-9　从饼图到面包圈图

很多时候，面包圈图的中心都用于放置标签或者其他内容，就像图5-9一样。

提示　需要记住的是，饼图和面包圈图很容易就会变乱。它们本来就不适合应对较多数值。

现在让我们再创建一次这个图表，不过用的是另一款免费且开源的可视化工具Protovis。它是一个通过JavaScript实现的函数库，利用了现代浏览器的可缩放矢量图形（SVG，Scalable Vector Graphics）功能。图形可以动态创建，而且支持动画和交互行为，这让Protovis非常适合在线创建图表。

▶ 访问http://vis.stanford.edu/protovis/下载Protovis，把它解压后放在你存储示例文件的文件夹里。

虽然我们要用到的是另一种编程语言，但步骤和之前用R与Illustrator的时候完全一样。首先载入数据，然后生成基础图表，最后是美感上的改进。

图5-10显示了我们希望达到的效果。它和图5-9很类似，但标签都被设置了角度，而且当鼠标悬停在各弧形上时，可以看到相应的投票数量。Protovis可以支持更高级的交互行为，不过我们还是从基础开始吧。

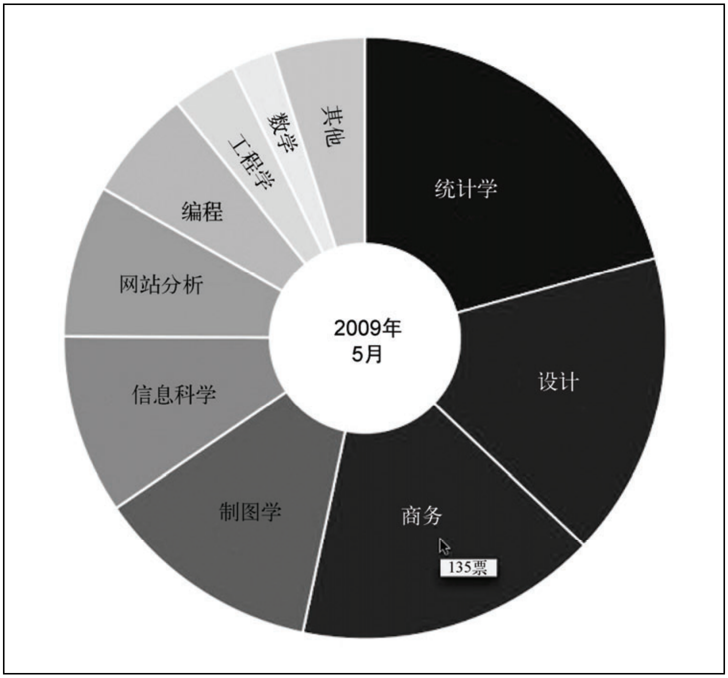


图5-10　利用Protovis创建的面包圈图

你要做的第一件事是创建一个HTML网页，将其命名为donut.html。

```
<html>
<head>
    <title>Donut Chart</title>
    <script type="text/javascript" src="protovis-r3.2.js"></script>
    <style type="text/css">
        #figure {
            width: 400px;
            height: 400px;
        }
    </style>
</head>
<body>
    <div id="figure">
```

```
    </div><!-- @结束figure -->
</body>
</html>
```

如果你曾经创建过网页，那么以上代码是很容易看懂的。如果你没有相关经验，这段代码其实在网上随处可见，你可以轻松找到。每一个网页都会以<html>标签开始，它的后面是<head>标签，里面包含有关于页面的信息，但不会显示在浏览器窗口中。只有<body>标签中封闭的所有内容才是可见的。将页面的标题（title）设置为Donut Chart，并通过<script>标签载入Protovis函数库，也就是JavaScript文件。之后再指定一些CSS，它们被用于定制HTML页面的样式。在这里简单一点就行了，将id为figure的<div>的宽度和高度都设定为400像素。你将会在它里面绘制图表。以上的HTML代码并不是图表的组成部分，但却是必需的，因为只有有了它，后面的JavaScript才会正确地载入到浏览器中。现在用浏览器打开donut.html文件，你看到的只是一个空白的页面。

在id为figure的<div>里，指定你要写的代码是JavaScript。后面的代码都位于这两个<script>标签之内。

```
<script type="text/javascript+protovis">
</script>
```

现在走好第一步：数据。我们关注的依然是FlowingData网站的投票结果，你需要将它们存储在数组中。一个数组用于存储投票数量，另一个用于存储相应的类别名称。

```
var data = [172,136,135,101,80,68,50,29,19,41];

var cats = ["统计学", "设计", "商务", "制图学",
    "信息科学", "网站分析", "编程",
    "工程学", "数学", "其他"];
```

然后指定面包圈图的宽度、高度、半径长度以及弧长范围。

```
var w = 350,
    h = 350,
    r = w / 2,
    a = pv.Scale.linear(0, pv.sum(data)).range(0, 2 * Math.PI);
```

面包圈图的宽度和高度都是350像素，半径（图表中心到外围的距离）是宽度的一半，也就是175像素。第4行指定了弧形的标度。这样来理解它：真正的数据是一个线性比例，从0到所有投票的总和，也就是总票数。然后把这个比例换算成面包圈的比例，即0～2π的弧度，如果你习惯按角度来考虑，那么也就是0～360°。

之后是设定颜色的变化。某个类别获得的票数越多，它的红色就应该越暗。在Illustrator里面你必须手动去调色，但Protovis可以自动为你选择颜色。你只需定义想要的颜色范围即可。

```
var depthColors = pv.Scale.linear(0, 172).range("white", "#821122");
```

现在你设定了颜色从白色到暗红色（也就是#821122）。这是一个线性范围，从0到172，后者是单个类别的最高票数。换句话说，票数为0的类别会显示为白色，票数为172的类别会显示为暗红色。票数在这个范围内的各类别会显示为白色和暗红色之间的某个颜色。

5

到目前为止，我们定义的都是一些变量，分别指定了尺寸和标度。要想得到实际的图表，首先需要创建一个350像素宽、350像素高的空白区域（panel）。

```
var vis = new pv.Panel()
     .width(w)
     .height(h);
```

然后往这个区域中添加内容，在本例中就是楔形。以下代码可能会让你有些困惑，我会逐行来解释。

```
vis.add(pv.Wedge)
     .data(data)
     .bottom(w / 2)
     .left(w / 2)
     .innerRadius(r - 120)
     .outerRadius(r)
     .fillStyle(function(d) depthColors(d))
     .strokeStyle("#fff")
     .angle(a)
     .title(function(d) String(d) + "票")
     .anchor("center").add(pv.Label)
          .text(function(d) cats[this.index]);
```

第一行说的是你正在往空白区域中添加楔形，分别对应数组中的每一个数据。`bottom()`和`left()`参数指定了楔形的朝向，使它们的尖端放置在圆形正中。`innerRadius()`指定了圆心那个洞的半径，而`outerRadius()`则是整个圆形的半径。这两者确定了楔形的大小。这些代码指定了面包圈图的几何结构。

在填充颜色方面，并非给填充样式（fill style）指定一个静态色值，而是由各个数据的值以及存储了颜色范围的`depthColors`变量来决定。换句话说，填充色是由各个数据点的函数决定的。此外添加了白色（`#fff`）边框，由`strokeStyle()`指定。之前定义的弧长标度可以确定每一个楔形的角度大小。

我们希望在读者将鼠标移动到某个区间上时显示该类别获得了多少票数，因此这里用到了`title()`来显示提示文本。你也可以创建一个mouseover事件来指定当用户把鼠标移动到某个对象上时发生什么，但浏览器本来就自动支持显示`title`属性内的值，所以用`title()`会更省事一些。让每一个数据点的`title`显示该数据点的值，然后再加上“票”以让表意更加清楚。最后为每一个区间添加标签。现在只剩下在图表中心的洞里加上“2009年5月”了。

```
vis.anchor("center").add(pv.Label)
     .font("bold 14px Georgia")
     .text("2009年5月")
```

以上代码可以理解为：“在图表中心添加标签，字体是14像素的Georgia粗体，内容是‘2009年5月’。”

所有的元素都搭建完毕，现在你可以渲染图形了。

```
vis.render();
```

在浏览器里打开donut.html，你就会看到图5-10的效果。

▶ 访问http://book.flowingdata.com/ch05/donut.html 看一下图表的在线效果，并且查看完整的源代码。

如果你是编程新手，这一节的内容可能会让你感到有些气馁。但好消息是Protovis有很多实例教程可供学习。官方网站上有许多可运行的实例，其中既有传统的统计学图表，也有更高级的可交互式动画图表，而且你可以使用自己的数据。所以如果你感到有些沮丧，千万不要就此灰心丧气。你现在稍微付出一些努力，很快就能掌握其中的窍门，而且将会获得丰厚的回报。让我们在下一节再来看看Protovis。

5.2.3 比例中的堆叠

在上一章我们用堆叠柱形图呈现了随时间变化的数据，但它的能力并不只限于时间数据。如图5-11所示，你还可以用堆叠柱形图来表现类别数据。

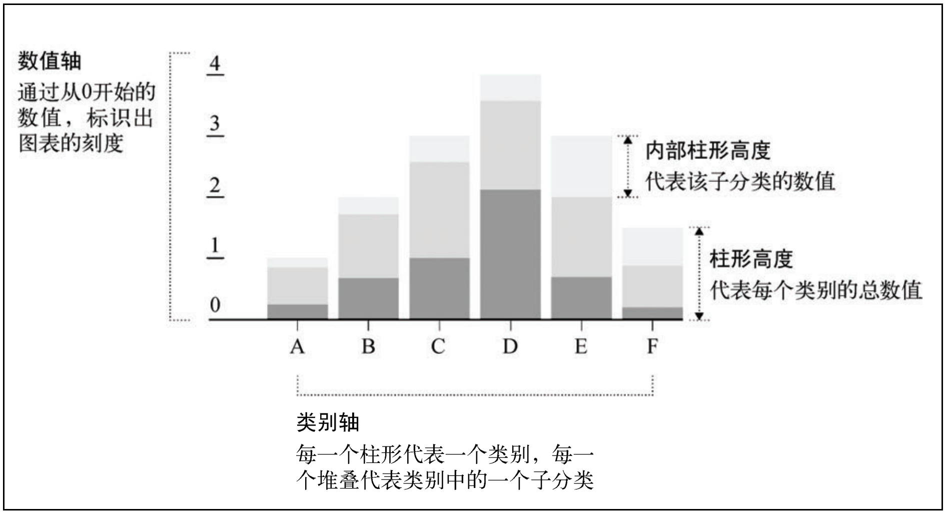


图5-11 按类别划分的堆叠柱形图

比如说，让我们看一下2010年7月和8月间由盖洛普公司[①]和哥伦比亚广播公司举办的一次有关奥巴马总统支持率的民意调查。参与者被问到他们是否支持奥巴马有关13个主要问题的政治举措。

① 盖洛普公司由乔治・盖洛普于1935年创立，是全球知名的民意测验和商业调查咨询公司。乔治・盖洛普（Gallup George Horace，1901—1984）是美国著名的数学家、社会科学家，抽样调查方法的创始人、民意调查的组织者，他几乎是民意调查活动的代名词。

以下是以表格形式给出的各个数字。

问　　题	支　　持	反　　对	不发表意见
种族关系	52	38	10
教育	49	40	11
恐怖活动	48	45	7
能源政策	47	42	11
外交事务	44	48	8
环境	43	51	6
伊拉克局势	41	53	6
税收	41	54	5
医疗保健政策	40	57	3
经济	38	59	3
阿富汗局势	36	57	7
联邦预算赤字	31	64	5
外来移民	29	62	9

我们可以为每一个问题创建一个饼图，如图5-12所示。要想用Illustrator实现，你只需输入多行数据即可。每一行数据都会生成一个饼图。

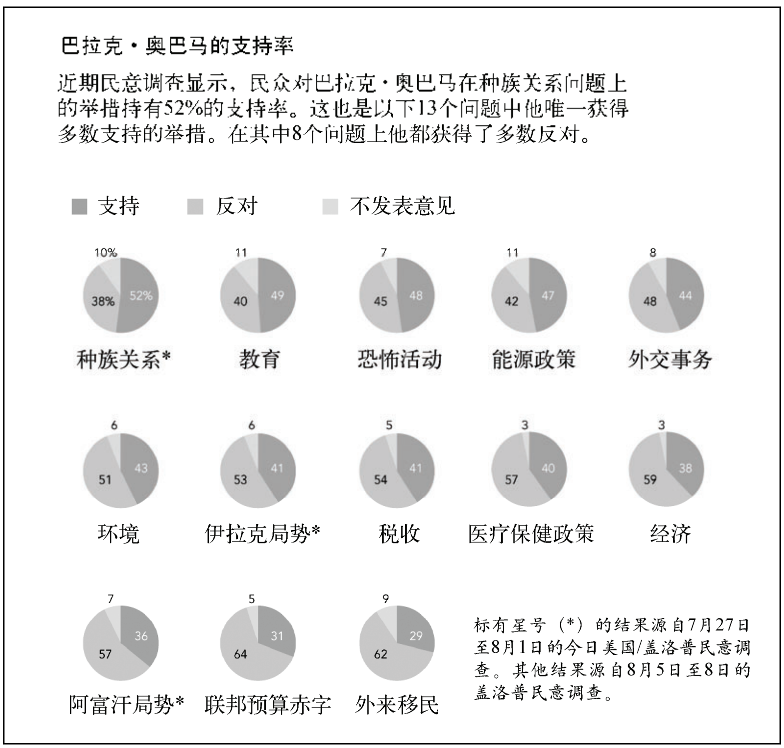


图5-12　系列饼图（另见彩插图5-12）

不过，堆叠柱形图会让你更容易地比较各个问题的支持率，因为与楔形的角度相比，人们对柱形高度更加敏感。所以让我们尝试一下。在Illustrator里面可以直接通过堆积柱形图工具（Stacked Graph tool）生成堆叠柱形图，这次我们再往里面添加一些简单的交互行为。

创建可交互堆叠柱形图

和面包圈图一样，我们也用Protovis来创建可交互堆叠柱形图。图5-13显示了最终的效果。我们需要实现两个基础的交互行为：第一个是当鼠标悬停在某个堆叠上时显示它的百分比值，第二个是根据鼠标的位置高亮显示该类别所对应的所有支持、反对或者不发表意见的柱形。

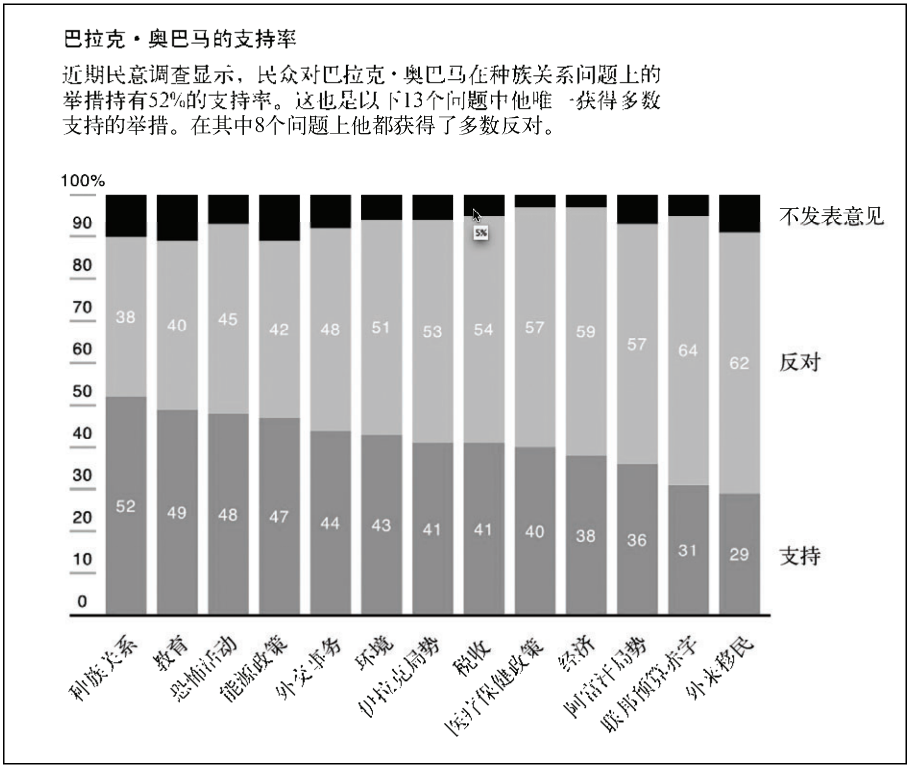


图5-13　利用Protovis创建的可交互堆叠柱形图（另见彩插图5-13）

一开始先创建HTML页面，并且载入必需的Protovis JavaScript文件。

```
<html>
<head>
    <title>Stacked Bar Chart</title>
    <script type="text/javascript" src="protovis-r3.2.js"></script>
</head>
<body>
```

```
        <div id="figure-wrapper">
            <div id="figure">

        </div><!-- @结束figure -->
        </div><!-- @结束figure-wrapper -->

</body>
</html>
```

这段代码应该很眼熟，和用Protovis创建的面包圈图时所做的一样。唯一的区别在于页面的标题变成了“Stacked Bar Chart”，而且增加了一个id为figure-wrapper的`<div>`。我们也没有添加任何CSS样式，这个可以留在后面去做。

现在开始写JavaScript。在id为figure的`<div>`中，以数组的形式载入数据（也就是奥巴马的支持率）。

```
<script type="text/javascript+protovis">
    var data = {
        "Issue":["种族关系","教育","恐怖活动","能源政策","外交事务","环境","伊拉克
        局势","税收","医疗保健政策","经济","阿富汗局势","联邦预算赤字","外来移民"],
        "支持":[52,49,48,47,44,43,41,41,40,38,36,31,29],
        "反对":[38,40,45,42,48,51,53,54,57,59,57,64,62],
        "不发表意见":[10,11,7,11,8,6,6,5,3,3,7,5,9]
    };
</script>
```

从中可以看出，对应种族关系的支持率和反对率分别是52%和38%。与之类似，对应教育的支持率和反对率分别是49%和40%。

为了便于编写图表部分的代码，你可以把数据分开存储为两个变量。

```
var cat = data.Issue;
var data = [data.Approve, data.Disapprove, data.None];
```

`Issue`（问题）数组被存储到`cat`变量中，而`data`变量则是一个包含数组的数组。

为宽度、高度、尺度比例和颜色设定必要的变量，代码如下：

```
var w = 400,
h = 250,
x = pv.Scale.ordinal(cat).splitBanded(0, w, 4/5),
y = pv.Scale.linear(0, 100).range(0, h),
fill = ["#809EAD", "#B1C0C9", "#D7D6CB"];
```

图表会有400像素宽，250像素高。水平轴的标度是有顺序的，表示你设定的各个类别，它们之间并非延续性的关系。类别也就是调查中涉及的那些问题。图表宽度的4/5用于柱形，其他空间用于各柱形间的间隔。

垂直轴表现的是百分比，是一个0 ~ 100%的线性标尺。各柱形的高度可以从0像素一直到图表的顶端，也就是250像素。

最后，填充色通过一个16进制的数组来指定。深蓝色代表支持，浅蓝色代表反对，浅灰色代表不发表意见。你也可以按自己的喜好改用其他颜色。

> ▶ 如果你不知道应该用什么颜色，http://colorbrewer2.org网站上的ColorBrewer可能会有所帮助。在该工具内指定需要的颜色数量和颜色类型，它就能提供适当的色标，方便你以多种格式进行复制。http://0to255.com上的0to255综合性更强，但我主要还是用前者。

下一步，用已指定的宽度和高度进行可视化的初始化。之后的代码为实际图表提供了周围的空白空间，以便添加坐标轴标签。比如说，`bottom(90)`将水平轴往上提升了90像素。你可以把这个操作看作是设置空白的画布。

```
var vis = new pv.Panel()
    .width(w)
    .height(h)
    .bottom(90)
    .left(32)
    .right(10)
    .top(15);
```

Protovis为堆叠图表提供了一种特殊的布局，其名称也很恰当，就叫做“堆叠”（Stack），方便在画布中添加堆叠柱形。虽然本例中添加的是堆叠柱形图，但该布局同样适用于堆叠面积图和流线图。将新的版式存储到`bar`变量中。


```
var bar = vis.add(pv.Layout.Stack)
    .layers(data)
    .x(function() x(this.index))
    .y(function(d) y(d))
    .layer.add(pv.Bar)
        .fillStyle(function() fill[this.parent.index])
    .width(x.range().band)
    .title(function(d) d + "%")
    .event("mouseover", function() this.fillStyle("#555"))
    .event("mouseout", function()
        this.fillStyle(fill[this.parent.index]));
```

另一条思路是把这个图表视作为三个层，分别代表支持、反对和不发表意见。还记得我们刚才是怎样把这三组数据构建为一个新数组的吗？用`layers()`调用它，而`x`和`y`依然保持你之前设定的标尺。

对每一个层用`pv.Bar`来添加柱形，用`fillStyle()`来指定填充色。请注意，我们使用了一个名为`this.parent.index`的函数。这样一来柱形就会根据它所属三个层中的其中一个进行着色。如果这里我们用的是`this.index`，你就必须为每一个柱形，也就是39（3×13）个柱形设置颜色。每个柱形的宽度都是相同的，具体数值可以从之前已指定的水平轴的顺序标尺中得到。

上面的最后三行代码给图表赋予了交互功能。在Protovis 里用`title()`就相当于给HTML元素（如图片）设置`title`属性。如果给网页中的图片设置了`title`，那么当鼠标悬停到图片上时就会出现文本提示。与之类似，当鼠标悬停到柱形上时也会出现文本提示。我们在这里只是简单地让文本提示显示该柱形所占的百分比值，其后跟随百分比符号（%）。

为了让鼠标移动到柱形上时高亮显示相应的层，我们用到了`event()`。`"mouseover"`时填充的是深灰色（`#555`），而当鼠标移出时，`"mouseout"`事件将各柱形恢复初始颜色。

提示　Protovis支持的交互行为不是仅限于鼠标悬停效果，你也可以设置诸如单击、双击等行为。查阅Protovis说明以了解更多细节。

最后你需要渲染图形以便生成图表。在JavaScript代码最后输入以下代码：

```
vis.render();
```

这行代码的意思是:好,我们已经准备好了所有的部件,现在绘制出图形吧。在浏览器(Firefox或Safari等现代浏览器）中打开网页，你就会看到类似图5-14中的图表。

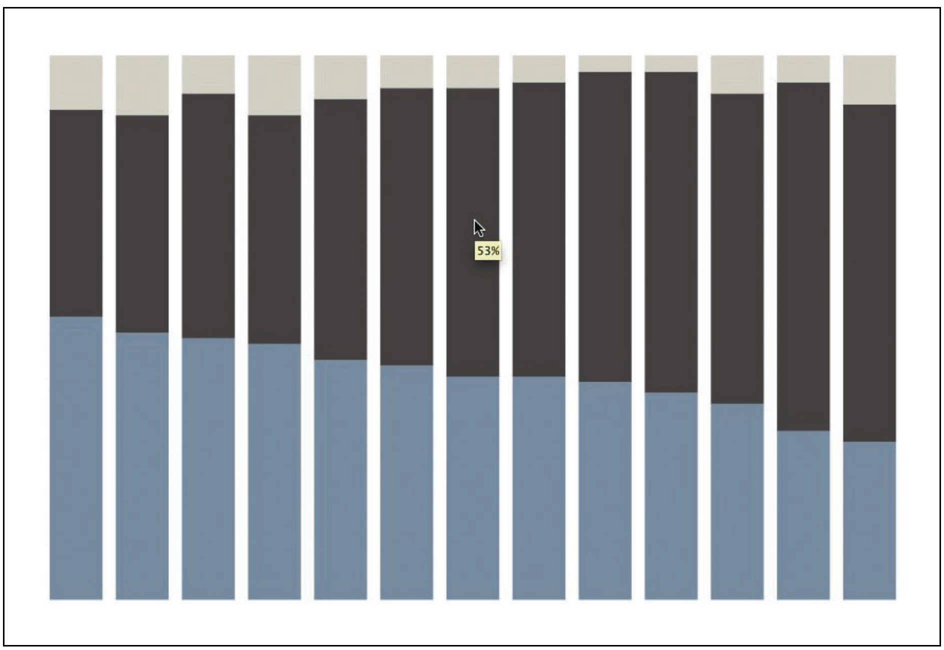


图5-14　没有添加标签的堆叠柱形图

鼠标移动到某个柱形上时，相应的层会被高亮显示，与此同时会出现文本提示。不过现在还缺少一些元素，也就是坐标轴和标签。把它们添加进来。

在图5-13中的柱形上有数字标签。不过标签只出现在较高的柱形上，灰色的柱形则没有。以

下是实现方法。请注意这些代码位于`vis.render()`的前面。永远都把渲染工作留在最后。

```
bar.anchor("center").add(pv.Label)
    .visible(function(d) d > 11)
    .textStyle("white")
    .text(function(d) d.toFixed(0));
```

检查每一个柱形是否大于11%。如果大于，那么就在柱形的正中间绘制一个白色的标签，它显示了所占百分比四舍五入后的整数数值。

现在为*x*轴上的每一个问题添加标签。理想情况是所有标签都水平显示，但很明显位置不够。如果我们的图表是横向的柱形图，那么标签倒是能够水平放下，不过在本例中，我们将它们以45° 斜向显示。当然你也可以把它们完全垂直放置，不过那样会有碍于阅读。

```
bar.anchor("bottom").add(pv.Label)
    .visible(function() !this.parent.index)
    .textAlign("right")
    .top(260)
    .left(function() x(this.index)+20)
    .textAngle(-Math.PI / 4)
    .text(function() cat[this.index]);
```

这段代码的原理和往柱形中间添加数字标签相同。不过，这次只会向位于底部的柱形添加标签，也就是代表支持的柱形。然后利用`textAlign()`和`top()`将文本右对齐，并设置它们的绝对垂直位置。它们的*x*轴位置是根据相应标签的柱形位置决定的。每一个标签都被旋转了45°，标签文本显示了类别的名称。

5

这样我们就有了类别标签。在垂直轴上添加数值标签也是同样的方法，不过你还需要添加刻度线。

```
vis.add(pv.Rule)
    .data(y.ticks())
    .bottom(y)
    .left(-15)
    .width(15)
    .strokeStyle(function(d) d > 0 ? "rgba(0,0,0,0.3)" : "#000")
    .anchor("top").add(pv.Label)
    .bottom(function(d) y(d)+2)
    .text(function(d) d == 100 ? "100%" : d.toFixed(0));
```

我们通过`y.ticks()`为图表添加标尺（Rule），也就是刻度线。零度线的颜色是黑色，其他刻度线为灰色。代码的第二部分在各刻度线的上面添加标签。

现在还差水平轴，所以还需要加入另一个标尺，如图5-15所示。

```
vis.add(pv.Rule)
    .bottom(y)
    .left(-15)
    .right(0)
    .strokeStyle("#000")
```

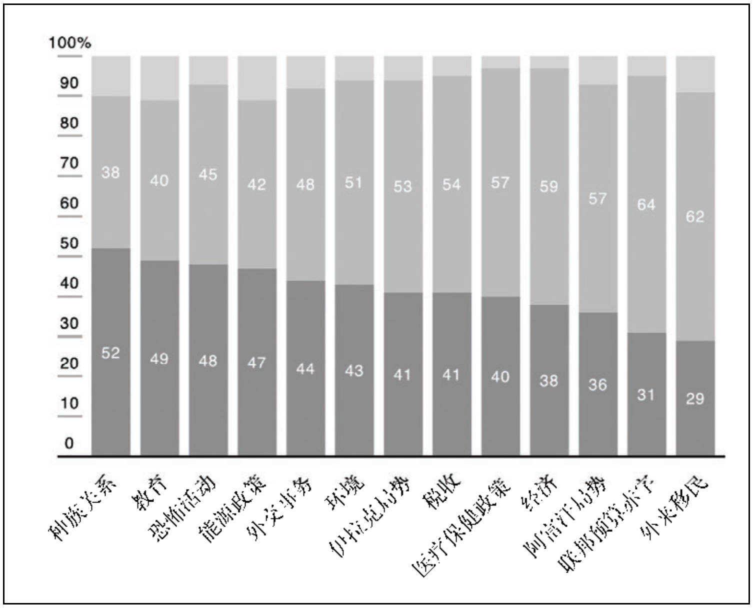


图5-15　添加水平轴

说明文字和其他标签是用HTML和CSS添加的。市面上有各种各样的Web设计书籍，所以这里就不再赘述了。Protovis是JavaScript 代码，因此与HTML和CSS可以轻松地无缝结合使用，这一点是最棒的。

▶ 访问http://book.flowingdata.com/ch05/stacked-bar.html浏览堆叠柱形图并与之交互，查看源代码了解HTML、CSS和JavaScript是如何紧密协作的。

5.2.4　层级和矩形

1990年，马里兰大学的Ben Shneiderman发现自己的硬盘总是空间不够，于是希望通过可视化的方式来弄清到底是什么如此占据空间。考虑到目录和文件是层级结构的，他一开始用的是树形图（tree diagram）。不过由于节点太多、分枝太多，很快图表就变得过于庞大，无法为他带来帮助。

▶ 访问http://datafl.ws/11m了解板块层级图的来龙去脉，以及由创造者Ben Shneiderman亲身描述的其他案例。

最后他的解决方案是板块层级图（treemap）。如图5-16所示，这是一种基于面积的可视化方式，通过每一个板块（通常为矩形）的尺寸大小进行度量。外部矩形代表父类别，而内部矩形代表子类别。你也可以用板块层级图显示单纯的比例关系，不过在充分利用该技术的情况下，它更适合于显示层级结构，或者更确切的说，树状结构的数据。

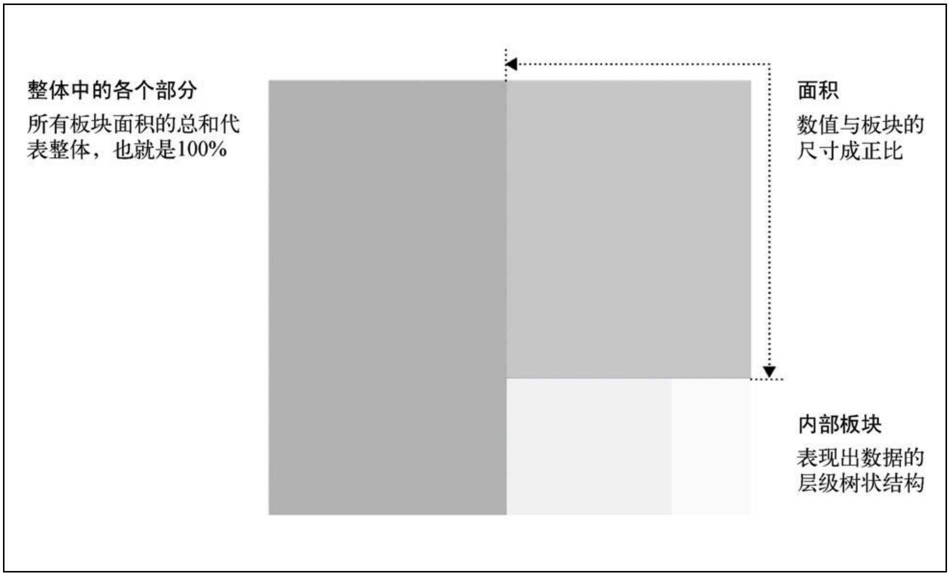


图5-16 板块层级图的基本框架

创建板块层级图

Illustrator没有提供板块层级图工具，但R中有一个工具包可以实现这一功能。这个工具包是由Jeff Enos和David Kane开发的，名叫Portfolio。它的开发初衷是对股票市场投资组合（stock market portfolio）进行可视化（其名由此而得），但也完全适用于我们自己的数据。让我们看看FlowingData网站上最受欢迎的100篇文章的浏览次数和评论数量，然后将它们按照文章类别进行划分，例如Visualization（可视化）或者Data Design Tips（数据设计贴士）。

提示 R是一款针对统计学计算的开源软件环境，你可以从http://www.r-project.org/免费下载。R的可贵之处在于它有一个活跃的社区，上面经常会有人开发新的工具包来添加功能。如果你需要创建静态图表，而又不知道从何开始，R将会是一个很好的起点。

和之前一样，第一步是在R中载入数据。你可以直接从计算机里载入，或者指定URL。本例中采用的是后者，因为数据可以直接在网上找到。如果你希望采用自己的数据、直接从计算机载入的话，请确保你的数据文件放在了R的工作路径中。你可以通过R中的“文件”菜单来修改工作路径。

通过URL来载入CSV文件非常容易，它只需要一行代码，用到了R中的`read.csv()`函数（参见图5-17）。

```
posts <- read.csv("http://datasets.flowingdata.com/post-data.txt")
```

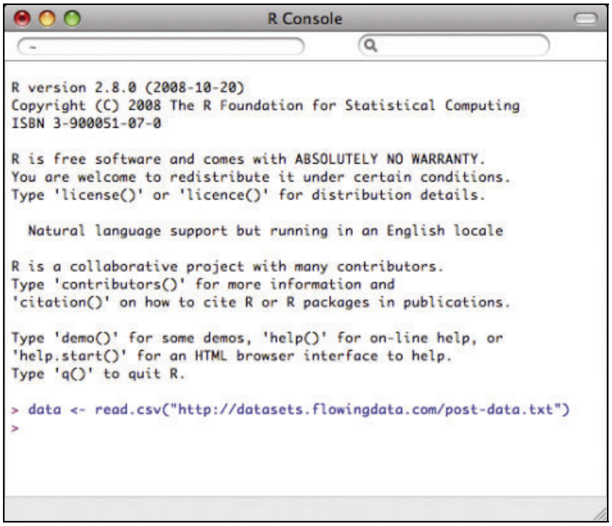


图5-17　在R中载入CSV文件

很简单，不是吗？我们已经利用`read.csv()`载入了一个文本文件（CSV格式），并且将页面的浏览量和评论数都存储进了一个名为`posts`的变量。正如上一章所提到的，`read.csv()`函数会假设你的数据文件是以逗号分隔的。假设你的数据是以制表符分隔，就需要用`sep`参数来指定该值为`\t`。要是你打算从本地目录中载入数据，之前的代码就应该变成这样：

```
posts <- read.csv("post-data.txt")
```

以上代码适用于你已经设置了相应工作路径的情况。要想深入了解如何利用`read.csv()`函数载入数据，在R的输入台中键入以下代码：

```
?read.csv
```

让我们继续。现在数据已经存储到了`posts`变量中，输入以下代码查看最开始的5行数据。

```
posts[1:5,]
```

你应该会看到数据有4列，分别对应原始CSV文件的id、views（浏览量）、comments（评论数）和category（文章类别）。既然R已经载入了数据，现在使用Portfolio工具包。输入以下代码来载入它：

```
library(portfolio)
```

返回了出错信息？你可能需要在使用之前先安装这个工具包：

```
install.packages("portfolio")
```

现在你应该可以载入工具包了。不再返回出错信息了吧？很好，让我们进入下一步。

提示　你也可以通过R的用户界面来安装工具包。在菜单中选择“工具包”（Packages）→“安装工具包”（Install Packages）。从工具包列表中找到需要的工具包然后单击“确定”按钮安装。

Portfolio工具包通过一个名为`map.market()`的函数来实现我们想要的效果。这个函数包含许多参数，不过我们只需要用到其中的5个。

```
map.market(id= posts $id, area=posts$views, group=posts$category,
    color=posts$comments, main="FlowingData Map")
```

名为`id`的列给每一篇文章指定了唯一的特征点，而你告诉R根据文章的浏览量来决定图表中各矩形的大小，根据文章类别进行分组，同时根据每篇文章的评论数量来决定矩形的颜色。最后，以FlowingData Map作为图表的主标题。敲回车键获得板块层级图，如图5-18所示。

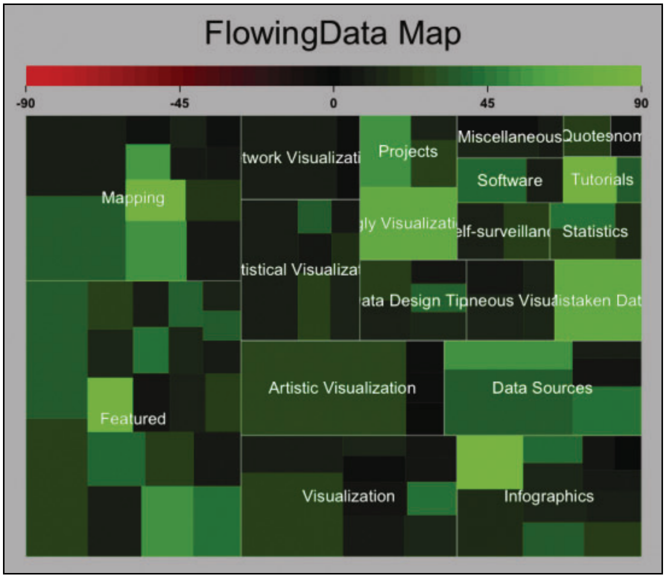


图5-18　R默认生成的板块层级图（另见彩插图5-18）

虽然图形看起来还很粗糙，但整个图表的基础和层级都已经搭建起来，这也是最困难的部分。和你指定的一样，每一个矩形都代表一篇文章，尺寸依页面浏览量而定，并按照文章类别进行了分组。明亮的绿色表明该文章获得的评论数比较多。浏览次数多的文章，获得的评论并不一定多。

在R中把图表存储为PDF文件，然后用Illustrator打开。利用各种常规选项来调整边框、填充色、字体，去掉无关的内容，并添加文字说明。

对于这个图表而言，我们需要修改一下标尺为–90 ~ 90的颜色说明。负值在这里并无意义，因为文章的评论数是不可能为负的。此外标签也需要调整，在较小的矩形中它们被遮盖住了。利用选择工具，根据矩形的尺寸（也就是该类别的浏览量）设置标签的字体大小。同时还可以加粗

各个类别的外边框，让它们更加明显。最后的效果应该如图5-19所示。

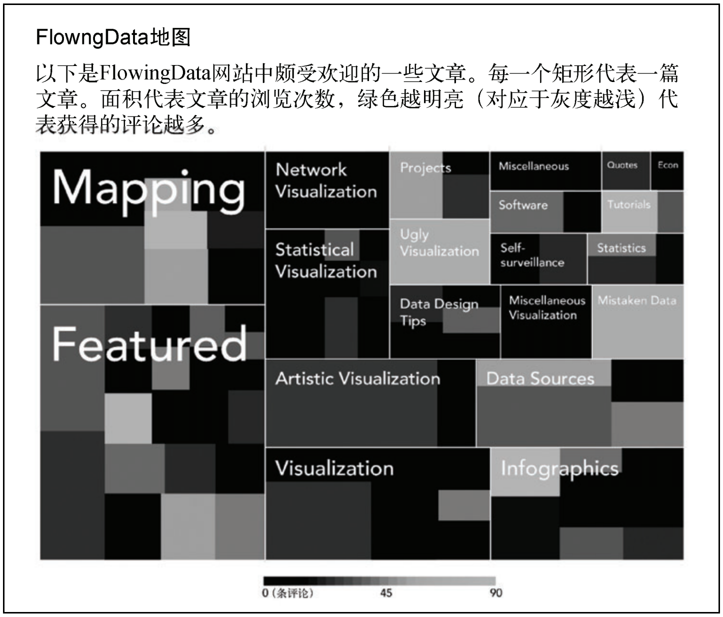


图5-19　在R中创建、在Illustrator中编辑后的板块层级图

就是这样。现在的各个标签更加明显，而且颜色说明的含义也更加精确，整个图表变得更加易读。另外，我们还去掉了深灰色的背景以提升简洁度。当然了，标题和文字说明会对阐明图表的主旨提供帮助。

▶ 在"How the Giants of Finance Shrank, Then Grew, Under the Financial Crisis"一文中，《纽约时报》使用了动态的板块层级图来显示金融危机时期股票市场的变化。地址是http://nyti.ms/9JUkWL。

Portfolio工具包在这一过程中承担了绝大部分工作。所以如果你打算采用自己的数据，唯一需要注意的地方就是确保正确的格式。你至少需要做到三件事：每一行数据需要一个独立的id、矩形尺寸的衡量依据、以及数据所属的父类别。你也可以用第四个度量标准来给矩形着色。第2章介绍了如何为数据定义必要的格式。

5.3　带时间属性的比例

我们也经常会遇到带有时间属性的比例。在一次民意调查中，每个问题都会有多个答案呈不同比例，而同样的民意调查有可能在一年中的每一个月都会举行。我们不仅关心单次调查的

结果，还希望看到大众的观点会如何随着时间而变化。一年前的调查结果和今天相比存在哪些差异？

当然，并不仅仅是民意调查会出现这种情况。各种比例分配都会随着时间而变化。在下面的例子中，我们来看看1860—2005年间美国人年龄结构的分布变化。随着卫生条件的改善和家庭平均人口的减少，整体的人口寿命与之前一代相比已经有了显著的提高。

5.3.1　堆叠的延续

假设你有多个时间序列图表，现在将它们从下往上堆叠，填满空白的区域。你得到的就是一个堆叠面积图，水平轴表示时间，垂直轴的范围为0～100%，如图5-20所示。

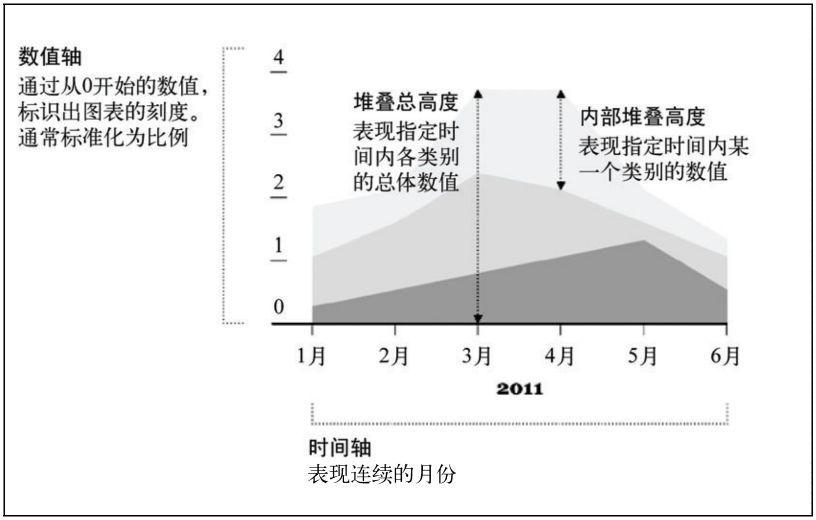


图5-20　堆叠面积图的基本框架

如果对这个面积图表进行垂直切片，就会得到该时间片段上的比例分布情况。或者你也可以把它看作是按时间相连的一系列堆积柱形图。

1. 创建堆叠面积图

本例中我们来看看人口的年龄结构。访问http://book.flowingdata.com/ch05/data/us-popula-tion-by-age.xls下载数据。在过去数十年中，医药和卫生保健都有了极大改善，人口的平均寿命持续提高，其结果是，中老年年龄段的人口比例出现了显著的上升。这些年里年龄分布到底发生了多大的改变？来自美国人口统计局的数据可以帮助你通过堆叠面积图得到以上问题的答案。你会看到中老年龄段的人口是如何增长，低年龄组的人口又是如何降低的。

有多种方法可以绘制这一图表，我们首先尝试用Illustrator。面积图工具（Area Graph Tool）可以帮助我们直接生成堆叠面积图（参见图5-21）。

在新文件中按住左键并拖动鼠标，在弹出的电子表格中输入数据。现在你已经熟悉载入数据、

生成图表、完善设计这一整套流程了，对吧？

在输入数据之后，你会看到一个堆叠面积图，如图5-22所示。

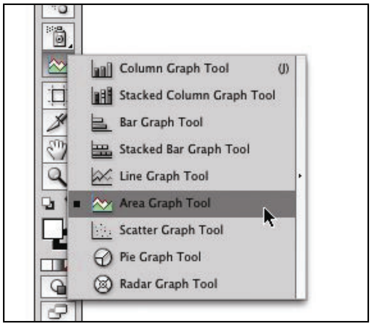


图5-21　面积图工具

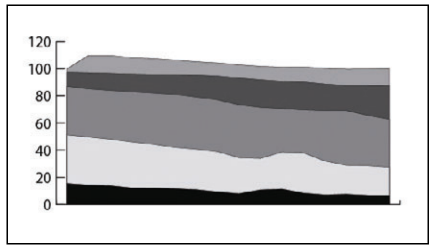


图5-22　Illustrator中默认生成的堆叠面积图

图5-22中的面积图超出了100%线。出现这种情况是因为堆叠面积图并没有采用标准化的比例，也就是一组累加等于100%的值。所以如果你希望每个时间段的比例总和是100%，就需要对原始数据进行标准化。图中的错误是我的责任，我把有些数据输错了，不好意思。修改过来之后，你看到的应该是图5-23中的样子。当然，你可能在第一次就正确地输入了数据，那么你就已经得到这一图表了。

不过还是要注意这类事情。最好能在一开始就检查出所有的笔误或者数据条目中的问题，不然等到设计完成之后再回过头来查找错误将是非常痛苦的事情。

提示　在手工输入数据时一定要小心。在数据的格式转换过程中，很多低级错误都可能发生。

现在基础图形已经没问题了，让我们去掉坐标轴和线条。使用直接选择工具选中想要的元素。我倾向于去掉垂直轴，然后留下较细的刻度线，以保持干净、轻巧的感觉。同时在数字后面加上百分百符号，这样表意更加明确。此外我通常都会把图形的边线从默认的黑色改为白色。最后为图形加入深浅不同的蓝色。效果如图5-24所示。

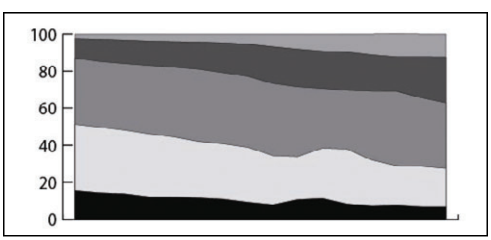


图5-23　修改后的面积图

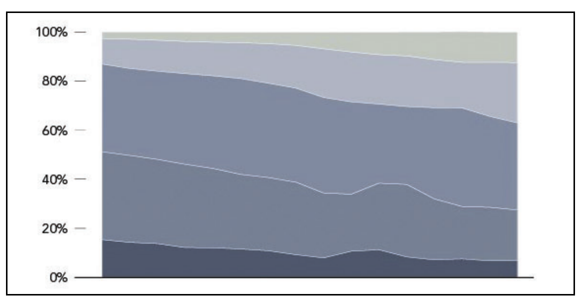


图5-24　修改后的颜色

再次重申，这只是我的设计品位，你可以按自己的喜好进行调整。颜色的选择同样也会受到图表主题的影响。你设计的图表越多，这方面的感觉就会越好。

提示　使用适合于主题的颜色，同时通过不同的色调来引导读者的视线。

是不是还少了点东西？嗯，水平轴上还没有标签。现在加上它们，并为各个面积区域添加标签，指明各个年龄段（参见图5-25）。

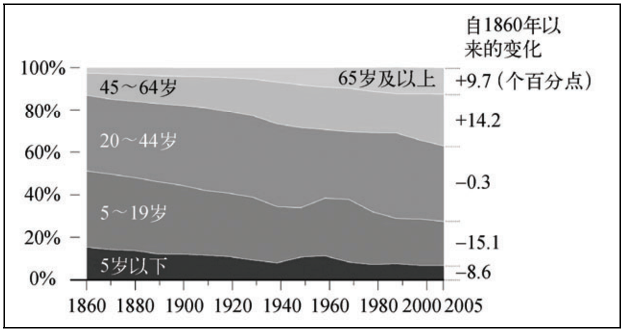


图5-25　添加标签后的堆叠面积图

我在图表的右侧也添加了注释。我们最感兴趣的是年龄段的变化，虽然可以从图表中了解到变化的趋势，但实际的数字更能说明问题。

最后，添加标题、文字说明，并在底部注明数据来源。稍微调整右侧注释的颜色，赋予更多的含义。现在我们得到了最终的图形，如图5-26所示。

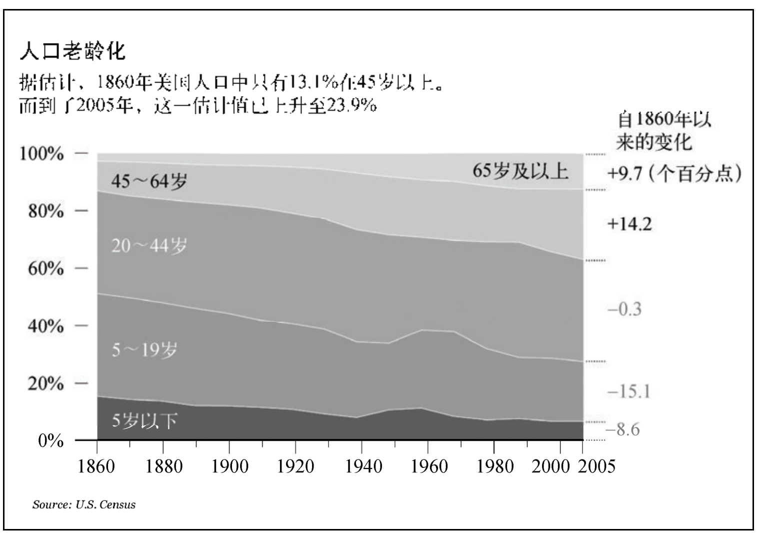


图5-26　最终的堆叠面积图

2. 创建可交互式堆叠面积图

堆叠面积图的缺点之一是当类别和数据点过多时，它们就会很难阅读，体现不出多少价值。在年龄段划分中这一图表类型很有效，是因为只有5个类别。如果类别继续增多，各层就会逐渐被压缩成细条。类似地，如果某个类别所占的数值比较小，那么它就很容易会被其他更加“粗壮”的类别挤得无处容身。不过，如果堆叠面积图是可交互的，这一问题就不存在了。

通过交互，读者可以自由搜索各类别，而且针对感兴趣的点可以缩放坐标轴进行细致的观察。文本提示可以帮助读者查看那些面积过小、无法放置标签的层的具体数值。简而言之，你可以将那些不适于静态堆叠面积图的数据应用于可交互式图表，从而便于读者浏览及探索。我们可以通过Protovis用JavaScript来实现，不过出于了解更多工具的目的（因为这一过程确实很有乐趣），这次让我们试一下Flash和ActionScript。

> ▶ Martin Wattenberg的NameVoyager让可交互式堆叠面积图广受欢迎。它会显示各时期婴儿的起名趋势，而且当你在搜索框内输入姓名后，图表会自动刷新。访问http://www.babyname-wizard.com/voyager进行体验。

说明　在线可视化目前已经开始从Flash向JavaScript和HTML5缓慢迁移，但并不是所有的浏览器都支持后者，比如说较早版本的Internet Explorer就不支持。此外，由于Flash已经流行很多年了，相比使用原生的浏览器功能，Flash的大量元件库和工具包能使任务变得更简单。

幸运的是，你不必一切都从零开始。通过由加州大学伯克利分校可视化实验室开发并维护的Flare可视化工具包，绝大部分工作都已经为你完成了。Flare是一个ActionScript库，实际上是一个名为Prefuse的Java可视化工具包的移植版本。我们会先看一下Flare网站上的一个应用案例JobVoyager，它和NameVoyager非常类似，但探索的是人们从事的职业。在你搭建好开发环境之后，剩下的就只是接入你自己的数据以及定制外观了。

说明　访问http://flare.prefuse.org/免费下载Flare，并解压到目标目录中。

你可以用ActionScript编写完整的代码，然后编译进Flash文件中。这意味着你先用自己理解的语言编写代码，然后通过编译器将代码翻译成计算机（或者说Flash播放器）能明白的比特编码，这样它才能服从你的命令。所以你需要两个东西：写代码的地方，以及编译的途径。

比较麻烦的做法是用某个标准文本编辑器来写代码，然后利用Adobe的免费编译器。我说它麻烦是因为其过程比较迂回，而且你必须先在计算机上安装很多东西。

比之更为简单、我鼎力推荐的做法是，如果你有不少项目都打算用到Flash和ActionScript，那么就使用Adobe的Flex Builder。它能把ActionScript编程中沉闷的部分加快解决，因为你写代码、编译和调试都在同一个地方。不过Flex Builder需要花钱购买（对学生是免费的）。如果你不确定是否值得花钱，也可以先下载免费试用版本，之后再作决定。对于我们这个堆叠面积图实例而言，下面将会解释在Flex Builder中的操作流程。

说明 在本书写作时，Adobe已经将Flex Builder改名为Flash Builder。这两者很相似，但也存在少许变化。以下步骤中虽然使用的是前者，但使用后者也是一样的操作[①]。访问http://www.adobe.com/products/flashbuilder/下载Flash Builder。确保能充分利用面向学生的折扣。只需提供你的学生证复印件，就能得到免费下载许可。或者你也可以找稍旧的Flex Builder版本，价格稍低。

当你下载并安装了Flex Builder之后，运行它，会看到一个窗口，如图5-27所示。

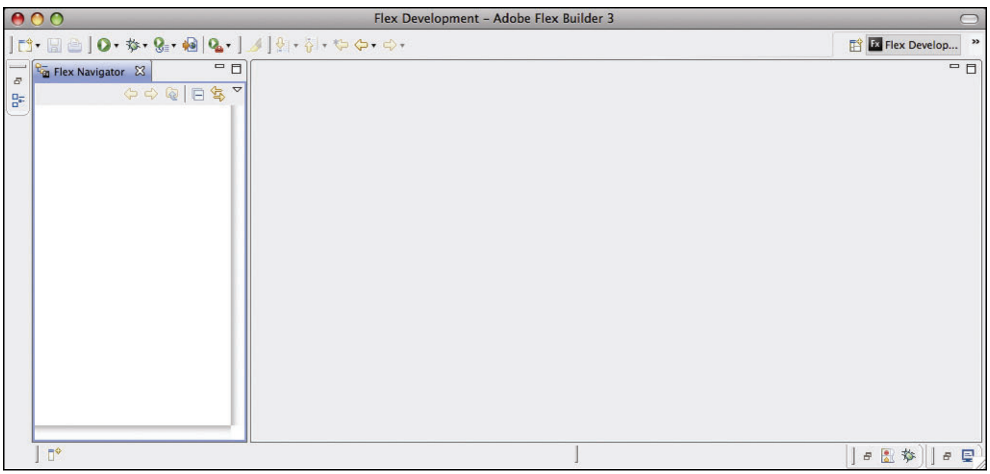


图5-27 运行Flex Builder后的初始界面

在左侧的Flex Navigator（包资源管理器）控件内单击右键，在弹出菜单中选择Import（导入）。你会看到一个弹出窗口，如图5-28所示。

选择General（常规）下的Existing Projects into Workspace（现有项目到工作空间中）并单击Next（下一步）按钮。在Select root directory （选择根目录）单选框处单击Browse（浏览）按钮找到你存放Flare文件的地方，并选择Flare目录，同时确保在项目窗口中的Flare已被选中，如图5-29所示。

① 目前的Adobe Flash Builder 4.5版本的界面已经和作者提供的配图不太一样，部分操作也有了变化。关于操作步骤，翻译过程中已经进行了相应的更改。

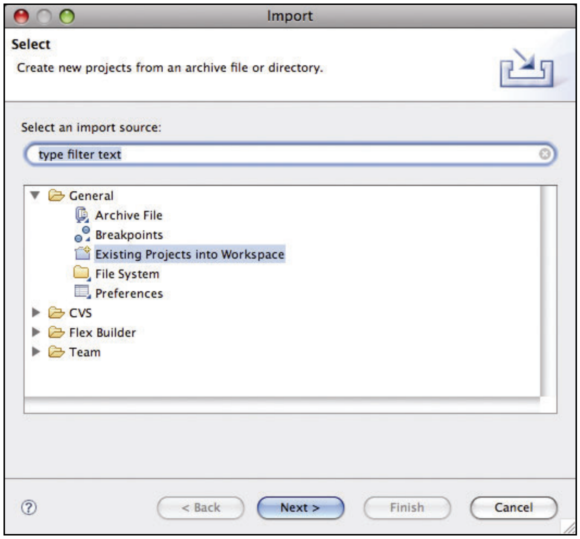


图5-28　Flex Builder中的Import窗口

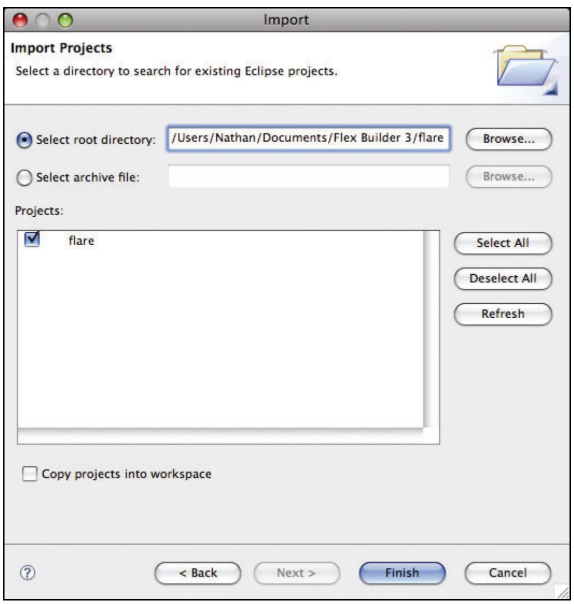


图5-29　“已有项目”窗口

单击Finish（完成）按钮。然后再重复一遍刚才的导入操作，这次选择的是flare.apps目录。在当前的包资源管理器中展开flare.apps/src/flare/apps/文件夹，并双击JobVoyager.as。现在你的Flex Builder窗口应该类似于图5-30。

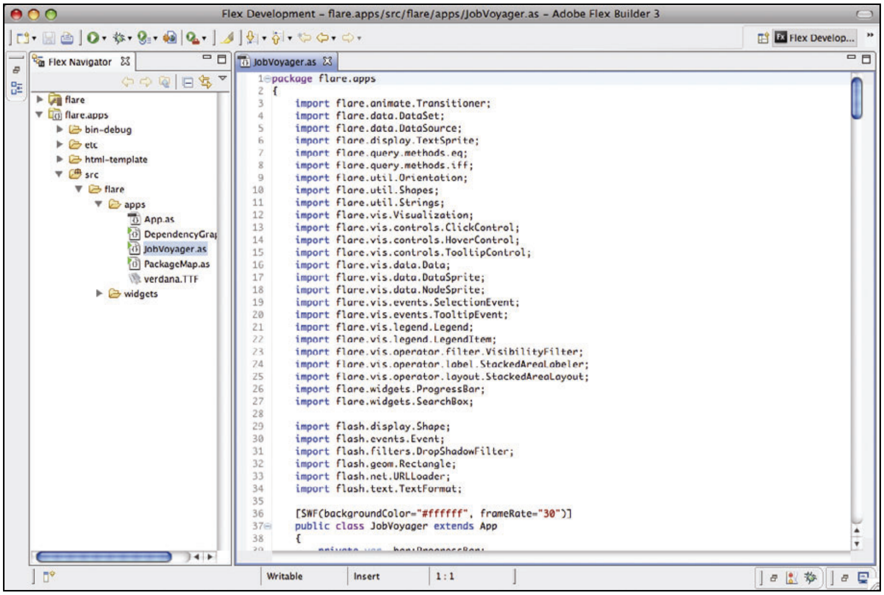


图5-30　打开后的JobVoyager代码

如果你现在单击运行按钮（界面左上角带有白色播放三角形的绿色按钮），你就会看到运行后的JobVoyager，如图5-31所示。这表示你已经完成了最困难的部分：安装。现在我们只需要插入自己的数据，然后按自己的喜好进行调整即可。这个过程听起来很熟悉吧？

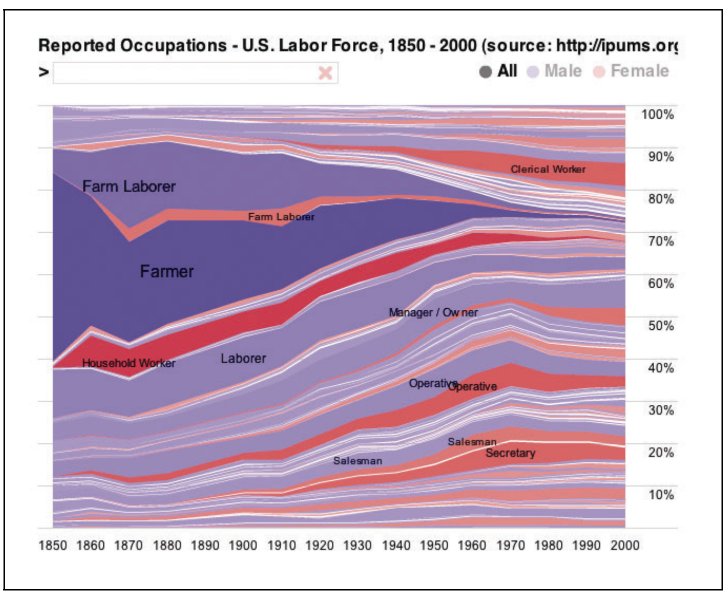


图5-31　JobVoyager应用（另见彩插图5-31）

图5-32显示了我们最后要实现的效果。它显示了1984—2008年间美国的消费开支状况，数据来源于美国人口统计局。水平轴显示的依然是年份，但图标表现的不是职业类别，而是开支分类，例如住房（Housing）和食品（Food）。

▶ 访问 http://datafl.ws/16r体验最终的可视化效果，并观察各种交互方式。

现在你需要改变数据源，它在JobVoyager.as的第57行进行了指定。

```
private var _url:String = "http://flare.prefuse.org/data/jobs.txt";
```

把`_url`改为指向我们提供开支数据的地址http://datasets.flowingdata.com/expenditures.txt 。和`jobs.txt`一样，这一数据也是以制表符分隔的文件。第一列是年份，第二列是类别，最后一列是消费额度。

```
private var _url:String =
    "http://datasets.flowingdata.com/expenditures.txt";
```

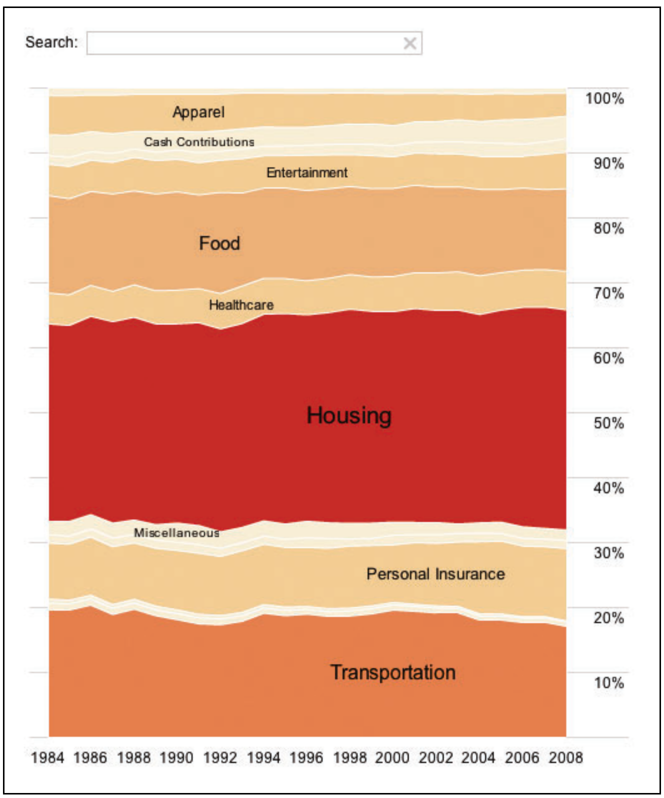


图5-32　有关消费开支的可交互式堆叠面积图（另见彩插图5-32）

现在文件将读取我们的消费开支数据，而不再是职业方面的数据了。到目前为止都还比较简单。

下面的第58和59两行是第一列的数据，在本例中也就是原始数据文件（jobs.txt）中列出的不同年份，是从1850年到2000年、以10年为单位进行的划分。你可以编写代码让程序自动在载入的开支数据中找到年份，不过由于这一数据不会变化，所以我们也可以节省点时间，直接明确指定数据的各个年份即可。

开支数据是从1984年到2008年、以年为单位进行划分的，所以我们把第58和59两行进行相应的修改。

```
private var _cols:Array =
    [1984,1985,1986,1987,1988,1989,1990,1991,1992,
    1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,
    2003,2004,2005,2006,2007,2008];
```

下一步修改涉及数据的标头。原始的数据文件（jobs.txt）共有4列：年份（year）、职业（occupation）、人数（people）和性别（sex）。开支数据中只有3列：年份（year）、开支类别（category）和消费额度（expenditure）。你需要让代码适应新的数据结构。

幸运的是，这很容易。年份列和之前是一样的，你只需要把人数改变为消费额度（对应垂直轴）、把职业改变为类别（对应各层）即可。最后，把所有用到性别的地方都删除。

第74行的作用是重塑数据，以便呈现堆叠面积图的形式。它将职业、性别两者指定为类别（也就是说，有职业和性别两种层），然后在*x*轴调用年份，在*y*轴调用人数。

```
var dr:Array = reshape(ds.nodes.data, ["occupation","sex"],
    "year", "people", _cols);
```

把代码改成这样：

```
var dr:Array = reshape(ds.nodes.data, ["category"],
    "year", "expenditure", _cols);
```

在我们的数据中，只会用到一个类别（没有性别层），也就是开支类别（category）。*x*轴依旧显示年份，而*y*轴是消费额度。

第84行将数据按照职业（按首字母顺序）和性别（以数字表现）排序。现在我们只需要按照开支类别排序：

```
data.nodes.sortBy("data.category");
```

你现在是否有点概念了？基本上所有代码都已经呈现在你眼前，你只需调整一些变量以适应新的数据即可。

提示 可视化领域中有很多优秀的开源项目。虽然现在你仍对编写代码“谈虎色变”，但在很多情况下可以借用这些项目中已有的代码来表现你自己的数据，你的工作只是调整一些变量而已。挑战在于读懂代码，理解它们是如何运作的。

第92行的作用是通过性别对各层进行着色，但我们并未通过性别来分割数据，所以不需要这样做。删除整行代码：

```
data.nodes.setProperty("fillHue", iff(eq("data.sex",1), 0.7, 0));
```

稍后我们会回头再调整各层的颜色。

第103行的作用是根据职业为各层添加标签：

```
_vis.operators.add(new StackedAreaLabeler("data.occupation"));
```

我们希望的是根据开支类别来添加标签，所以进行相应改动：

```
_vis.operators.add(new StackedAreaLabeler("data.category"));
```

第213～231行处理JobVoyager的过滤。首先设置的是男性/女性的过滤，然后是职业的过滤。我们不需要前者，所以可以删除第215～218行，然后使第219行成为一个简单的`if`语句。

与之类似，第260～289行（也就是删除`if`条件之前的第264～293行）创建了按钮来触发男性/女性的过滤。我们也可以删除它们。

现在我们基本上快把职业数据全部改为自己的开支数据了。回到第213行的`filter()`函数。像之前一样更新该函数，以按开支类别而不是职业过滤。

现在的第218行（也就是删除`if`条件之前的第222行）是这样：

```
var s:String = String(d.data["occupation"]).toLowerCase();
```

把occupation改为category：

```
var s:String = String(d.data["category"]).toLowerCase();
```

下一个需要调整的是颜色。如果你现在开始编译代码并运行，就会得到一个红色系的堆叠面积图，如图5-33所示。而我们需要让对比再增强一点。

颜色在两个地方指定。第86～89行指定了边框颜色，并且对所有元素都使用了红色：

```
shape: Shapes.POLYGON,
lineColor: 0,
fillValue: 1,
fillSaturation: 0.5
```

之后在第105行通过计数更新了色彩的饱和度（红色的级别）。其中`SaturationEncoder()`的代码在第360～383行。我们并不打算用不同的饱和度来区分各层，而是明确指定各层的颜色值。

首先，按以下代码更新第86～89行：

```
shape: Shapes.POLYGON,
lineColor: 0xFFFFFFFF
```

现在`lineColor`已经指定了边框色为白色。如果开支的类别比较多，可能不应该这样做，因为那样会显得比较杂乱。现在类别并不多，所以用白色边框可以提高图表的易读性。

下一步，建立一个颜色的数组，这些颜色是根据级别进行排序的。在第50行前加入以下代码：

```
private var _reds:Array = [0xFFFEF0D9, 0xFFFDD49E, 0xFFFDBB84, 0xFFFC8D59,
0xFFE34A33, 0xFFB30000];
```

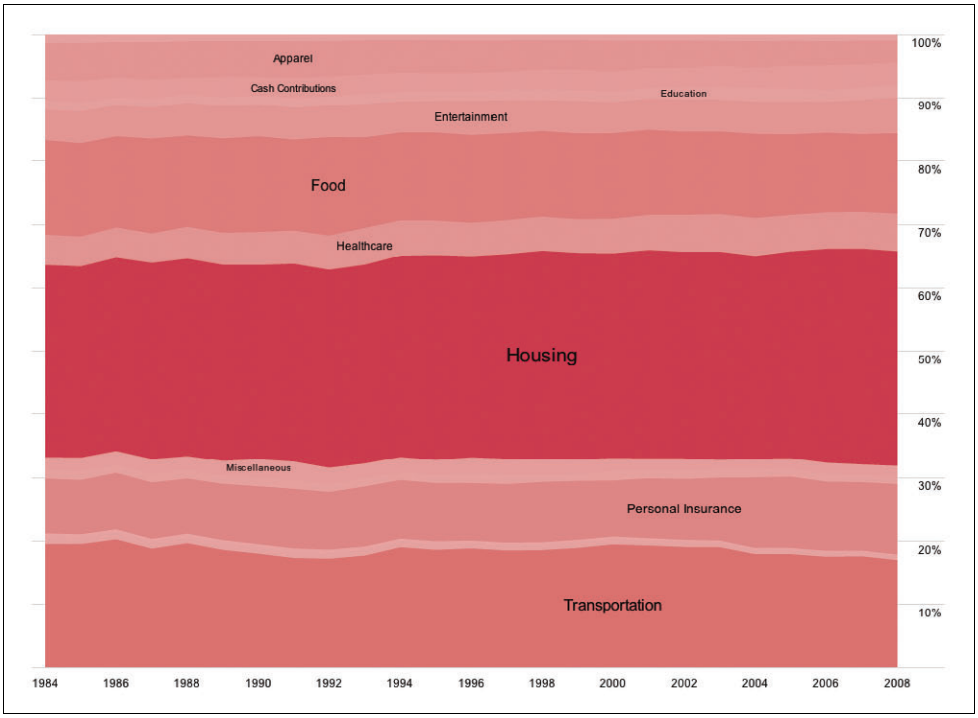


图5-33　基础色调的堆叠面积图（另见彩插图5-33）

我是用ColorBrewer（之前提到过）来找到这些颜色的，该工具能根据我设置的条件提供用色建议。它本身是用于地图选色的，但对于一般的可视化也很适用。

现在在第110行添加一个新的ColorEncoder：

```
var colorPalette:ColorPalette = new ColorPalette(_reds);
vis.operators.add(new ColorEncoder("data.max", "nodes",
    "fillColor", null, colorPalette));
```

说明　如果在编译代码时出现错误，看看JobVoyager.as文件的前面部分是否指定了以下两行用于导入ColorPallete和Encoder对象的代码。如果没有的话，添加进去。

```
import "are.util.palette.*;
import "are.vis.operator.encoder.*;
```

啊哈！现在再运行应该就能得到我们想要的效果了（参见图5-32）。当然，你不必止步于此，还有很多事情可以做。你可以接入自己的数据，改变配色方案，或者进一步定制来满足自己的需求，比如改变字体或文本提示的格式。甚至你还可以和其他工具整合，或者加入更多的ActionScript代码等。

5.3.2　逐点详述

堆叠面积图也存在缺点，比如每一个层的变化趋势可能会难以识别，因为每一个数据点的位置都受到了它下方点的影响。所以有时候用直接的时间序列图来表现分布反而会更直观一些。

幸运的是，在Illustrator中这两者很容易转换。数据的输入方法是一样的，所以你只需改变图表类型即可。在人口年龄结构的问题中，如果一开始选择折线图工具而非面积图工具，你就会得到如图5-34所示的默认图表。

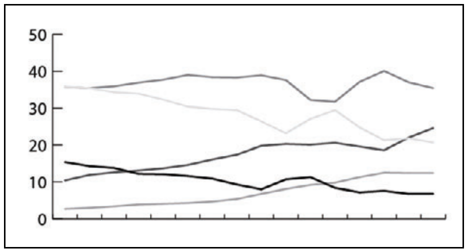


图5-34　默认的折线图

按之前的方法，根据自己的喜好对图表和格式进行调整、简化，这样针对同一数据我们就有了另一种视角（参见图5-35）。

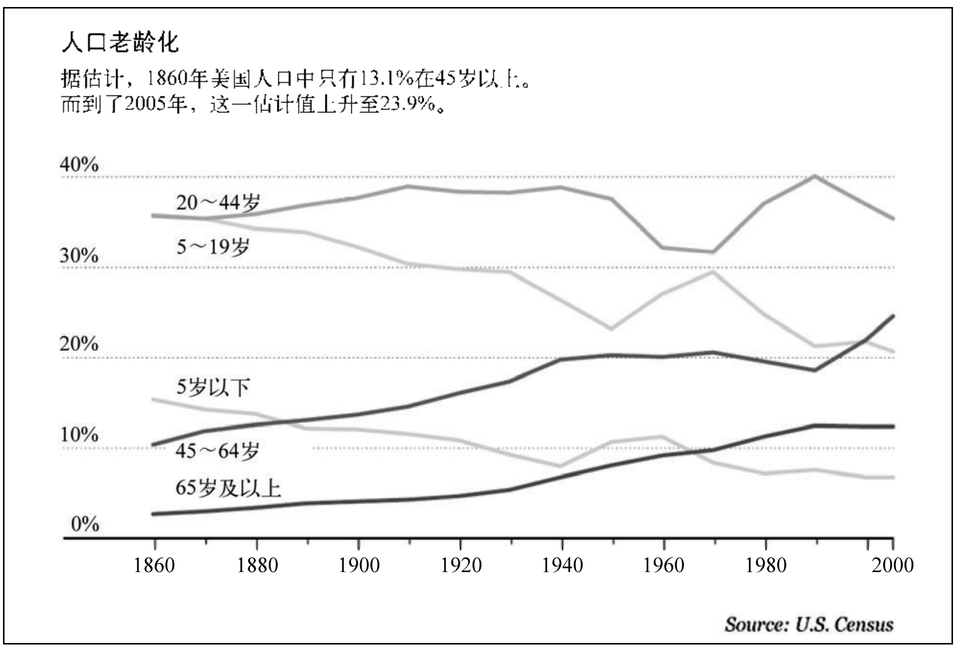


图5-35　简化并添加标记后的折线图

有了这个时间序列图表，更容易看出每一个年龄段各自的变化趋势。但是，图表丧失了整体感，也没有体现出其中的比例分布状况。选择何种图表应该反映出你希望表达的重点是什么。如果页面上有足够的空间，你甚至也可以把两种图都放上去。

5.4 小结

比例分布数据和其他数据类型的主要区别在于它们体现的是整体中的各个部分。每一个单独的数值都有其意义，而各部分相加得到的子集或者总和也是如此。我们的可视化成果应该体现出这些方面。

如果数据并不多，那么饼图可能最适合你。当然也可以用面包圈图，不过要稍加小心。如果有多个类别的多个数值，可以考虑堆叠柱形图，而非多个饼图。如果你想探索其中随时间而变化的模式，则可以使用堆叠面积图，或者用传统的时间序列图。有了这些作为基础，你的比例图一定会出类拔萃。

在设计和实施阶段，先自问你希望从数据中得到什么信息，然后以此为起点。依靠静态图表是否能把故事讲述完整？很多时候答案都会是“能”，这很好。但如果你决定要用可交互图表，那么请现在纸上拟定出各种交互行为，比如在单击对象时应该发生什么、不应该发生什么。如果你往界面中强塞太多功能，很快就会变得难以掌控。所以请尽量保持简单。找人来试着体验一下，看看他们是否能理解界面的操作。

最后，在代码阶段，你很可能会遇到下一步不知该怎么办的窘境，对于编程新手来说尤其如此。我自己也总是碰到这种情况。当你卡壳时，互联网是最好的求助之所。找一下是否有合适的说明文档，或者与你手中项目相似的研究案例。不要只关注于语法，要学习其中的逻辑思路，这才是最能帮助你的。很幸运，像Protovis和Flare这些函数库都有很多实例，也提供了优秀的说明文档。

在下一章中，我们将继续深入分析、演绎数据，同时拉上我们的好朋友统计学。在研究数据集与变量之间的关系时，R会继续提供很多帮助。准备好了吗？让我们开始吧。

6 有关关系的可视化

统计学就是要寻找数据之间的关系。两个群体彼此之间存在哪些共同点？群体内部的子群体之间有哪些共同点？子群体内部的个体之间又有哪些共同点？对于统计来说，人们最熟悉的关系就是关联性（correlation）。比方说，如果全民的平均身高增高了，那么平均体重自然也会随之增长。这是一种简单的、成正比的关联性。然而就和现实生活中一样，一旦我们开始考虑更多因素，或者试图寻找非线性的模式，那么数据间的关系就会变得更加复杂。本章讨论的是如何利用可视化的方法挖掘出并强调这些关系，从而讲出好故事。

在本章和下一章中，我们开始处理更加复杂的统计图表。你会发现R能带来极大的帮助，而这些地方正是这一开源软件的强大之处。和之前一样，R包揽了最基础、最繁重的工作，而Illustrator则能让生成的图表更加易读。

6.1　在关系中寻求什么

到目前为止，我们已经了解了有关时间和比例的一些基础数据关系。我们可以掌握对象随时间变化的趋势，或者比较对象中包含的各个部分或百分比，从而了解其中最大、最小和夹在中间的各个部分。接下来我们将探求不同变量间的关系。当某个数量增加时，另一个数量是否会减少？它们之间是因果关系还是关联关系？通常来说，前者很难通过数量进行证实，因此也难以用图表描述。但是关联性是很容易表现的，而它也能引导我们进行更加深入的探究和分析。

你也可以后退一步，从整体上进行观察，或者摸清数据的分布。群体彼此之间是存在交叠还是互不干涉？这种比较可以用来讲述有关一国国民的故事，也可以用来描述我们身边的人。我们可以了解不同国家之间的差异，或者世界发展的总体趋势，从而有利于制定人道援助或其他方面的决策。

你也可以从更宽泛的角度来比较各个分布数据。某个族群的组成是如何随时间发生变化的？或者它是如何保持不变的？

最重要的是，当图表最终都展现在你的眼前时，请自问这些结果的意义何在。它们是否在你的意料之中？有没有什么结果让你感到惊讶？

这看起来似乎颇为晦涩抽象，所以让我们直接通过具体实例来看看究竟应该如何观察数据之间的关系。

6.2　关联性

一想到数据的关系，人们的脑子里最先蹦出的概念可能就是关联性，之后便是因果关系。此刻你可能正在琢磨关联性和因果关系到底有什么差别。关联性意味着当一件事情变化时，另一件事情也可能会发生某种变化。比如说，每加仑牛奶的价格和每加仑汽油的价格就是正相关的（positively correlated）。它们都在逐年上升。

下面是关联性和因果关系之间的区别。如果你提升了汽油价格，牛奶价格是否会自动上升呢？更重要的是，如果牛奶的价格确实上升了，真的是汽油价格提升引起的吗？还是有其他外在因素，例如乳制品业突发的罢工行为？

解释所有外在的、混杂的因素无疑非常困难，因此证实因果关系也并非易事。研究人员可能要花费数年的时间来弄清楚这些事情。但是我们可以轻易地发现事物间的关联，而它也一样很有价值。以下各节将会帮助大家理解这一点。

关联性可以帮助我们根据某一已知指标来预测另一指标。要想探究这种关系，让我们看看散点图和多重散点图。

6.2.1　更多的圆点

在第4章中，我们曾用散点图来表示随时间而变化的指标：时间由水平轴表示，数值或量度则在垂直轴上表现。它有利于我们辨识出随时间发生的变化（或者不变化）。这种图表体现的是

时间和另一因素或变量之间的关系。其实，散点图不仅可以应用于时间，还可以表现两个变量之间的关系，如图6-1所示。

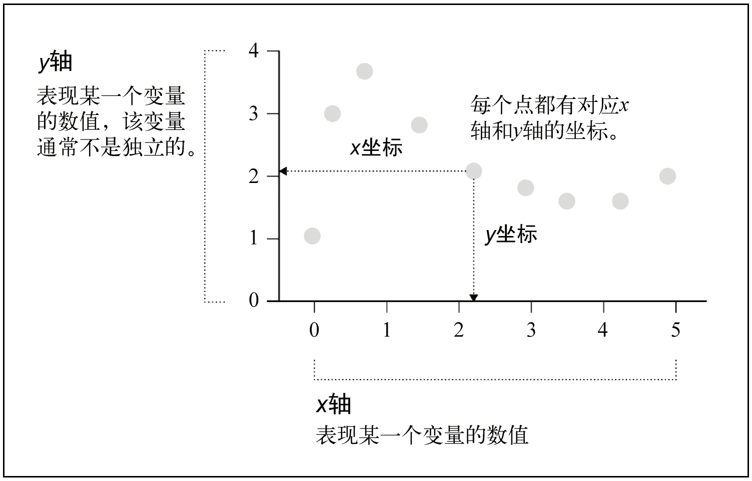


图6-1　散点图的基本框架，比较两个变量

如果两个指标是正相关的（参见图6-2左图），那么从左往右读图表时，点的位置就会越来越高。相反，如果是负相关，那么点的位置就会从左往右越来越低，如图6-2中图所示。

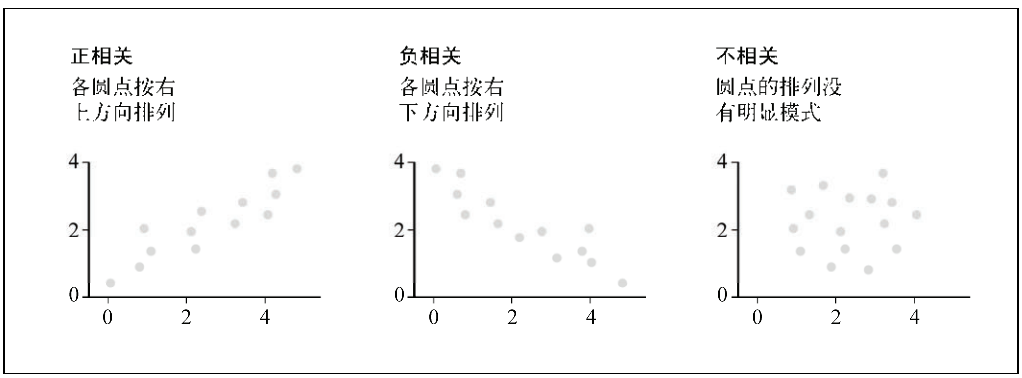


图6-2　散点图中显示的关联性

有时候数据之间的关系非常直接，例如人的身高和体重之间的关联性。通常，一个人的身高增加后体重也会随之增加。还有些时候关联性并没有那么明显，比如健康和体质指数（Body Mass Index，BMI）之间的关系。BMI过高通常意味着某人超重了，但肌肉发达的人（比如运动员）也会有很高的BMI。如果抽样群体是健美运动员或者橄榄球员又会是什么情况？健康和BMI之间有什么关系？

请记住图表只是整个故事的一部分而已。对故事结果的解读仍然取决于人。而这对于关系来

说非常重要。也许你会尝试表现出数据间的因果关系，但大多数情况下它们都并不准确。汽油价格和世界人口都在逐年增长，但这并不意味着降低汽油价格，人口增长就会减缓。

创建散点图

在本例中，我们看一下2005年美国各州的犯罪率，即美国人口统计局公布的每10万人口中谋杀、抢劫和故意伤害等罪案的发生率。总共有7种犯罪类型，让我们先来了解其中的两种：入室盗窃和谋杀。这两者之间是否存在联系？是否在谋杀率相对较高的州，入室盗窃也发生得较多？让我们用R来进行调查。

和之前一样，第一件事就是在R中利用`read.csv()`来载入数据。你可以访问http://datasets.flowingdata.com/crimeRatesByState2005.csv下载CSV文件，不过这里我们直接通过URL读取。

```
# 输入数据
crime <-
    read.csv('http://datasets.flowingdata.com/crimeRatesByState2005.csv',
    sep=",", header=TRUE)
```

现在检查一下头几行数据，输入变量`crime`，之后设置你希望看到的行数。

```
crime[1:3,]
```

以下是R返回的前几行数据。

```
          state murder forcible_rape robbery aggravated_assault burglary
1 United States    5.6          31.7   140.7              291.1    726.7
2       Alabama    8.2          34.3   141.4              247.8    953.8
3        Alaska    4.8          81.1    80.9              465.1    622.5
  larceny_theft motor_vehicle_theft population
1        2286.3               416.7  295753151
2        2650.0               288.3    4545049
3        2599.1               391.0     669488
```


第一列显示的是州名，其他列是各种类型犯罪的发生率。比如说，2005年美国全国的平均抢劫案发生率是每10万人口中发生140.7起。我们用`plot()`来为谋杀和入室盗窃两个数据创建默认的散点图，结果如图6-3所示。

```
plot(crime$murder, crime$burglary)
```

看上去这两个数据似乎是正相关的：谋杀率相对较高的州，其入室盗窃率也相对较高。但由于有一个点出现在很远的最右侧，所以这个关系并不是很容易发现。这个点（也就是离群值）导致水平轴必须延伸得很长。它代表的是华盛顿特区，该地区的谋杀率远高于其他地区，达到了35.4。谋杀率第二高的州是路易斯安那和马里兰，为9.9。

为了让图表更加清楚，带来更大的帮助，我们可以拿掉华盛顿特区，同时去掉全美平均值，聚焦于各个单独的州。

```
crime2 <- crime[crime$state != "District of Columbia",]
crime2 <- crime2[crime2$state != "United States",]
```

第一行在`crime2`中存储了华盛顿特区以外的数据。与之类似，第二行过滤掉了全美平均值。

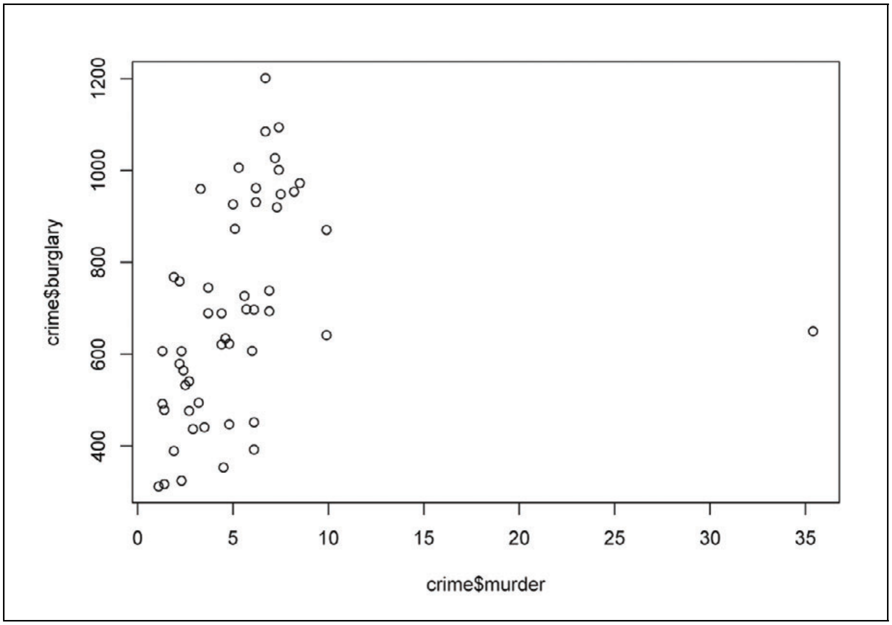


图6-3　默认生成的有关谋杀案和入室盗窃案的散点图

现在再次绘制得到的图表会更加清晰，如图6-4所示。

```
plot(crime2$murder, crime2$burglary)
```

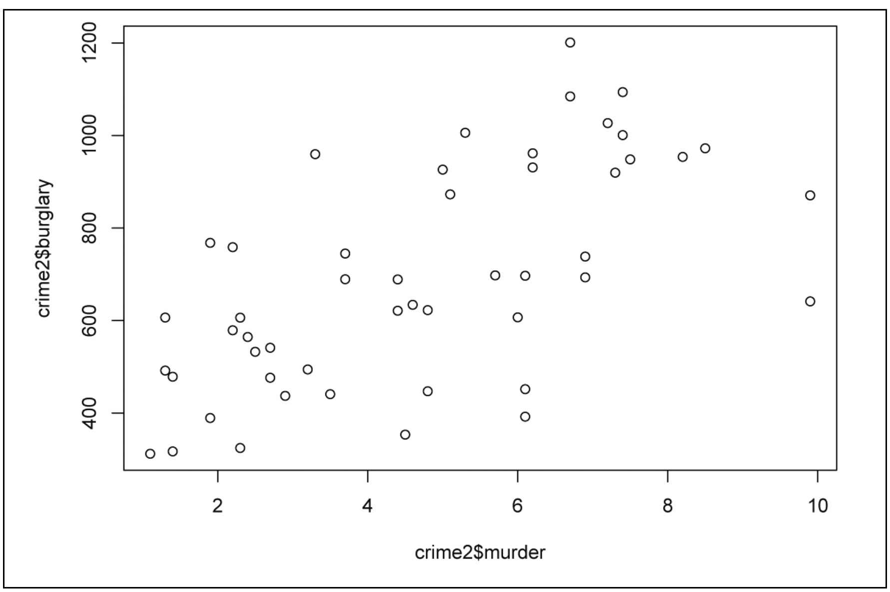


图6-4　过滤掉数据之后的散点图

如果让轴从0开始可能会更好，所以在这里再进行一些设置。*x*轴应该从0到10，*y*轴应该从0到1200。这样做会把所有点的位置向右上移动，结果如图6-5所示。

```
plot(crime2$murder, crime2$burglary, xlim=c(0,10), ylim=c(0, 1200))
```

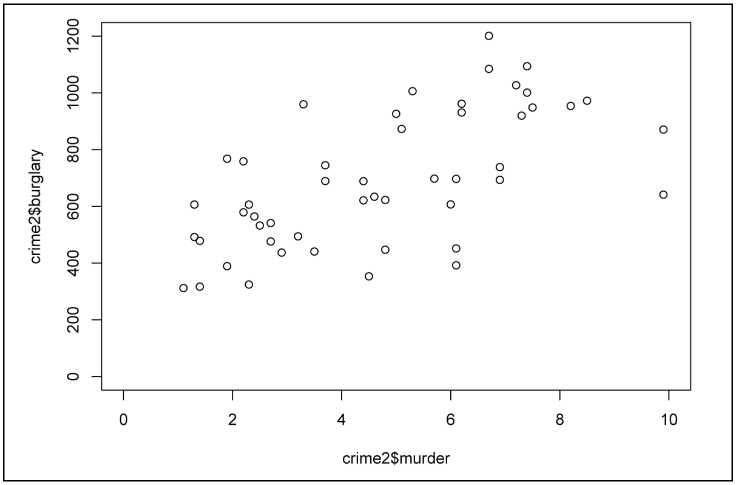


图6-5 坐标轴从0开始的散点图

你知道什么能够让这个图表更加有用吗？自然是第4章中提到的LOESS曲线。它能帮助我们更加明确地观察入室盗窃率和谋杀率之间的关系。通过`scatter.smooth()`来绘制各点间的曲线，结果如图6-6所示。

```
scatter.smooth(crime2$murder, crime2$burglary,
    xlim=c(0,10), ylim=c(0, 1200))
```

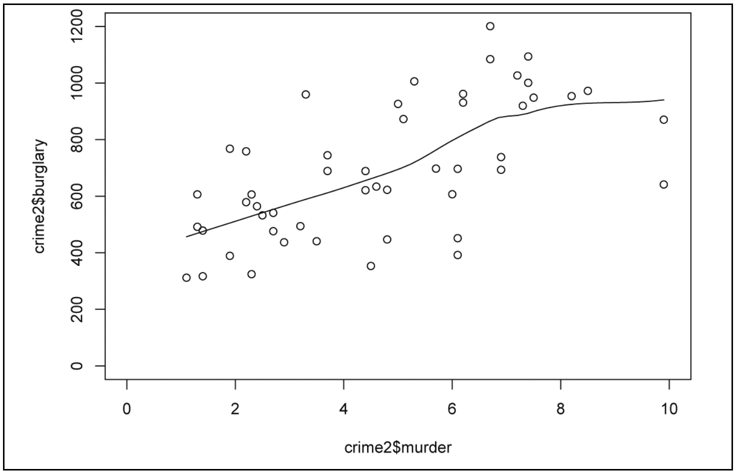



图6-6 运用曲线来估算关系

说明　为了简化起见，华盛顿特区被从数据集中移走了，这样便于我们更好地观察其他数据。但是数据中的这些离群值其实是很重要的。我们将会在第7章中继续讨论这一问题。

作为一个基础图表，这样已经很不错了。如果你只是为了方便分析，到底就算大功告成。但如果这份图表还会有其他读者，你也可以稍微作一些修改，改善它的可读性，如图6-7所示。

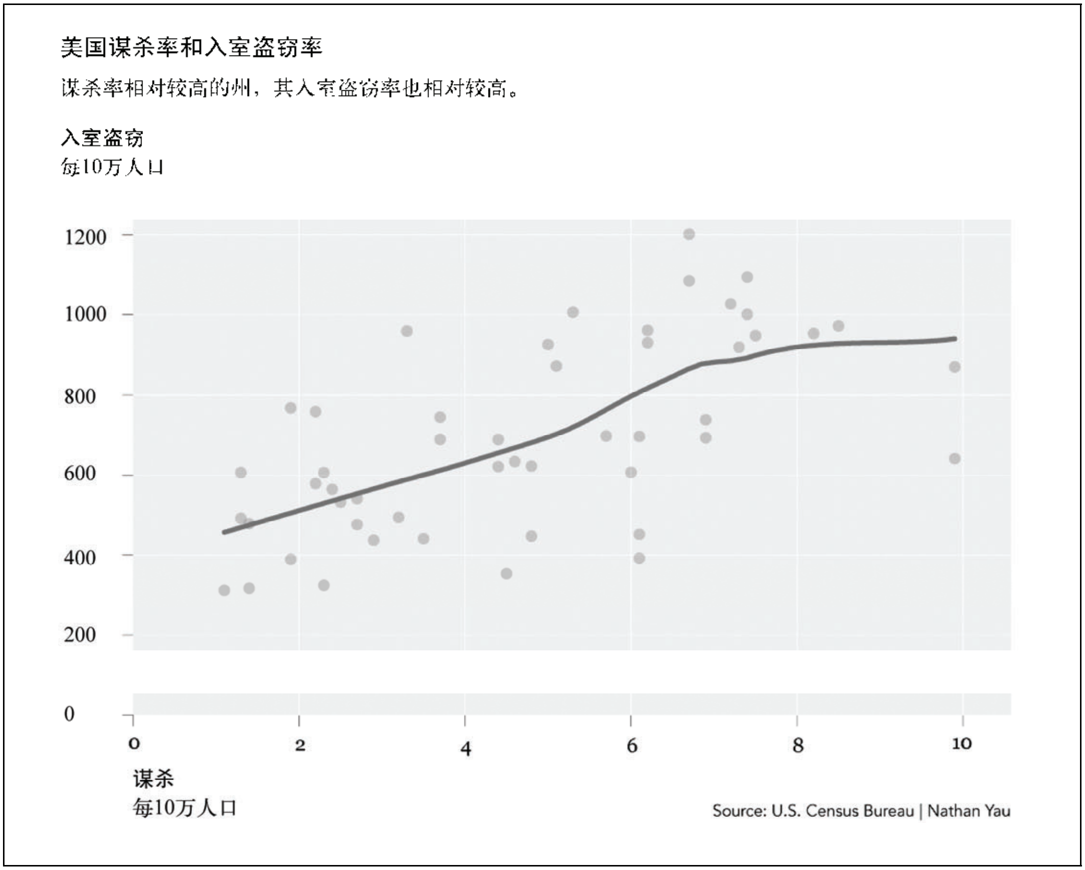


图6-7　改进后的谋杀案和入室盗窃案发生率散点图

我去掉了厚重的边框，削弱了四周的框架感，并加粗了曲线、将点的颜色调暗，从而将读者的注意力吸引到曲线上面。

6.2.2　探索更多的变量

现在我们已经针对两个变量进行了比较和绘制，下一步自然是比较其他变量。当然，你也可以总是只挑选两个变量，然后绘制它们的散点图，但这样一来很可能会丢掉很多机会，无法体现

出数据中其他有意思的地方。这是我们所不希望看到的。所以最好是用散点图矩阵的形式，绘制出每一种可能的配对，如图6-8所示。

这种方法在数据探索阶段特别有用。也许在你面前有一堆数据集，但你对从哪开始却没有任何头绪。如果你自己都不太清楚数据中蕴涵的意义，那么你的读者就难理解了。

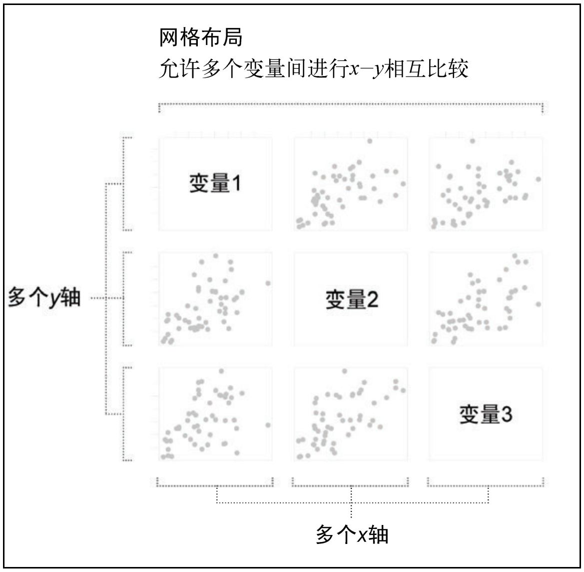


图6-8　散点图矩阵的基本框架

提示　必须理解自己的数据才能讲述一个完整的故事。你对数据了解得越透彻，所讲的故事就会越精彩。

散点图矩阵能够满足我们的期望。它通常是一个方格网，在垂直和水平方向上都列出了所有的变量。水平轴上的每一列都代表一个变量，垂直轴上的每一行也代表一个变量。这样就提供了所有可能的配对，而对角线则可以空出来放置标签，因为不用拿变量与其自身进行比较。

创建散点图矩阵

现在让我们回到犯罪率数据。我们有7个变量，或者说7个类型的犯罪率，但在之前的例子中我们只比较了其中的两个：谋杀案和入室盗窃案。而通过散点图矩阵，我们可以比较所有的犯罪类型。图6-9显示了最终的效果。

和我们预料的一样，有很多都是正相关。比如说入室盗窃和故意伤害的关联性就比较高。前者增加时后者也会随之增加，反之亦然。不过，谋杀和扒窃之间的关系就不是这么显而易见了。因此你不能轻易作出任何假设，而散点图矩阵的价值就体现在这里。有可能第一眼看上去你会感觉这些线条和图表很容易混淆，但从左到右、从上到下地阅读它，无疑会得到很多信息。

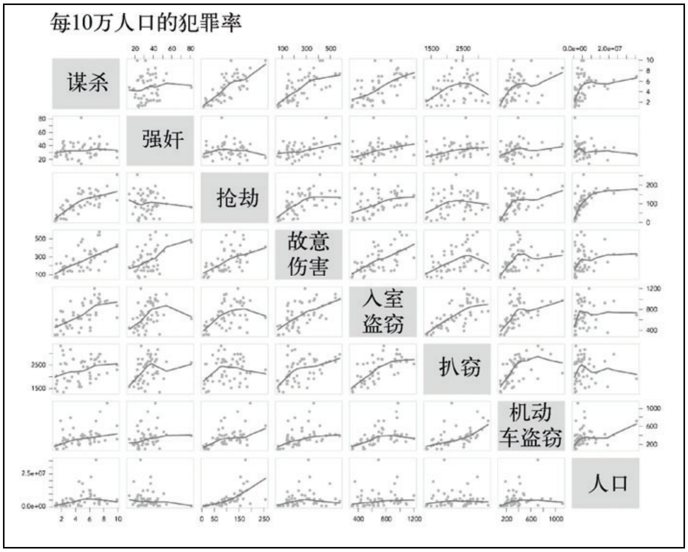


图6-9　美国各州犯罪率的散点图矩阵

很幸运，在R中创建散点图矩阵和创建单个的散点图一样容易。仍然还是利用`plot()`函数，不过这次不止传递两个列，而是整块数据，只不过减去第一列，因为它的内容是各个州名。

```
plot(crime2[,2:9])
```

这行代码能生成图6-10中显示的矩阵，和我们想要的已经不远了。不过要想帮助读者更容易地辨识数据间的关系，还应该加上拟合曲线。

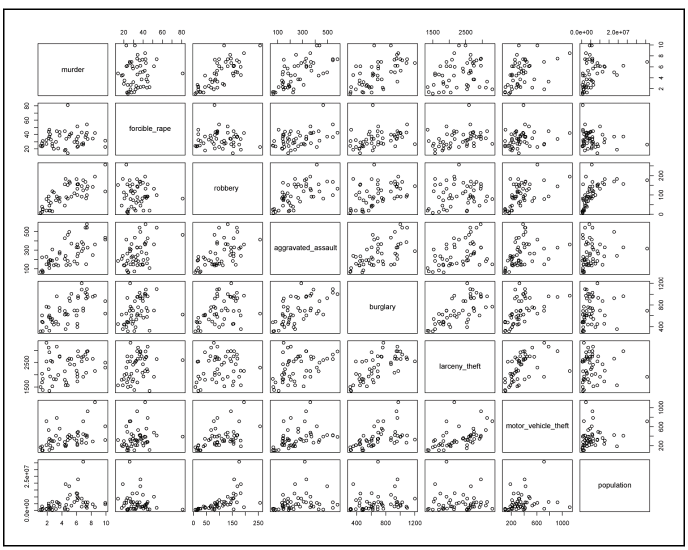


图6-10　R默认生成的散点图矩阵

要想创建带有拟合LOESS曲线的散点图矩阵，我们可以转而使用`pairs()`函数，其实也非常简单。结果如图6-11所示。

```
pairs(crime2[,2:9], panel=panel.smooth)
```

提示　`pairs()`中的`panel`参数调用了一个有关*x*和*y*值的函数，本例中也就是`panel.smooth()`。它是R中的一个原生函数，用于生成LOESS曲线。当然你也可以指定自己的函数。

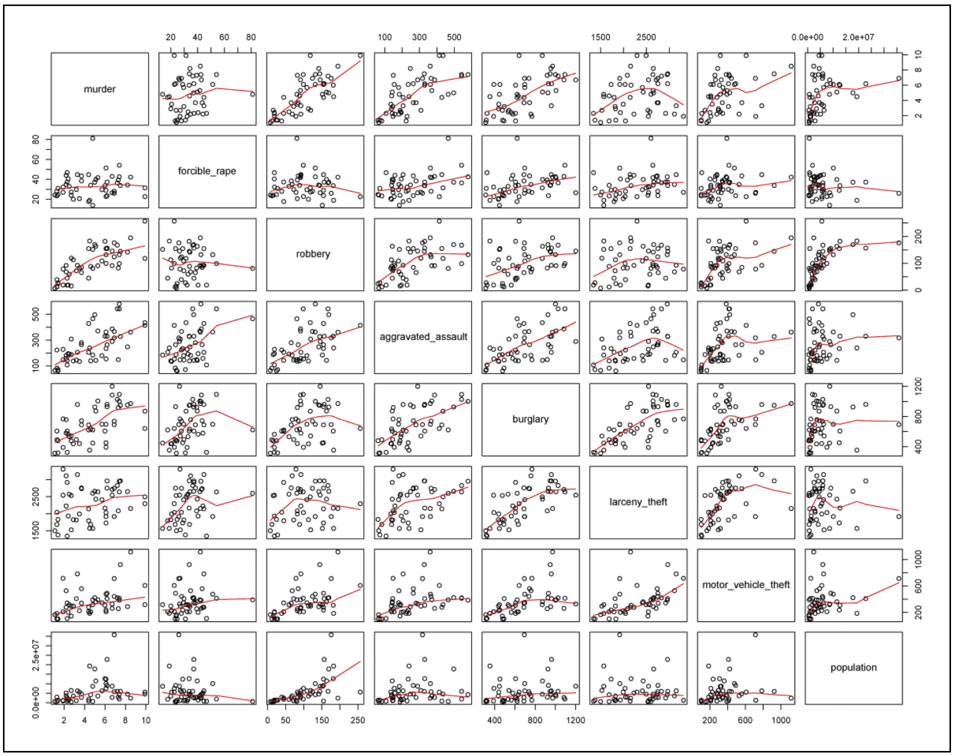


图6-11　带有LOESS曲线的散点图矩阵

现在我们已经有了一个很好的基础框架了，之后的工作就是让它提高可读性（参见图6-9的最终效果）。将图表存为PDF格式，然后在Illustrator里打开。

大体上来讲，我们需要削弱杂乱感，让重要的内容得到强调。犯罪类型和趋势曲线的重要性是第一位的，各个圆点其次，坐标轴最后，因此在颜色和大小上也应该按此轻重顺序进行调整。对于对角线上的标签，可以适当增加文字的大小，并用灰色填充方框，让它们凸显出来。然后加粗各条趋势曲线，调整它们的颜色，加大线与点之间的反差。最后减小边框和网格线的粗细，并改为浅灰色，使之得以弱化。对比图6-9和图6-11，很明显前者的表现力要更强一些，是不是？

提示　明确你要讲述的故事的重点，然后在图表设计中有意强化这些地方。但请注意不要扭曲事实。

6.2.3　气泡

自从卡罗琳学院的国际卫生学教授、Gapminder基金会理事Hans Rosling用他的动态图表来演示有关国家贫富和健康的故事以来，在*x*–*y*轴上按比例显示的气泡就为世人所熟知，而且受到了极大的欢迎。虽然他的动态图表通过动画来展现随时间发生的变化，但我们一样可以创建它的静态版本：气泡图。

▶ 访问Gapminder网站www.gapminder.org欣赏Hans Rosling的著名演讲，其中包括BBC出品的纪录片*The Joy of Stats*，该片讲述了统计学的无穷乐趣。

最简单的气泡图就是一系列尺寸按比例显示的气泡，不过现在我们可以考虑它的变体，也就是带有“气泡”维度的散点图。

这种图表类型的优势在于它便于我们一次比较3个变量，如图6-12所示。一个变量是*x*轴，一个变量是*y*轴，而第三个则通过气泡的面积大小来体现。

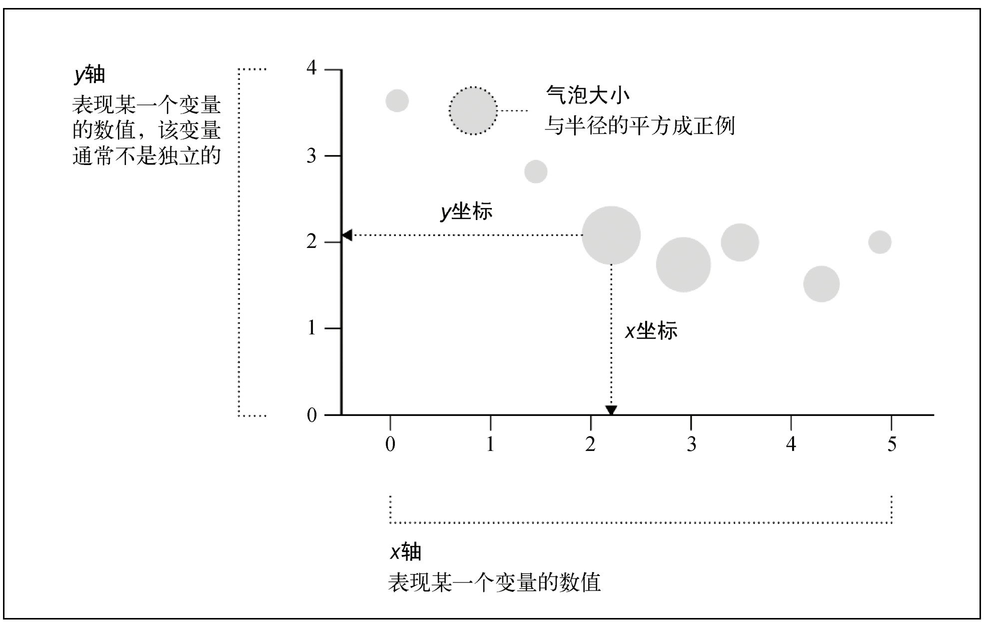


图6-12　气泡图的基本框架

需要注意的是气泡的面积大小，因为很多人会在这个地方出错。在第1章中我们就曾提到过，气泡的大小是根据面积来的，而不是半径、直径或者圆的周长。如果马虎地按软件默认选项进行绘制，最后得到的圆形就很可能不是过大就是过小。

提示 在用圆形表示数据时，要根据圆的面积来定义尺寸，而不是半径、直径或者周长。

我们举一个简单的例子来说明。假设你负责公司的广告销售，现在正在测试网站上的两个banner广告，以便了解哪一个广告的宣传效果更好。这两个广告都投放了一个月的时间，但获得的点击次数是不同的：第一个banner被点击了150次，而第二个banner只被点击了100次。因此，第一个banner的效果比第二个要好50%。图6-13中根据圆形的面积表现了两个banner气泡的大小关系，其中第一个圆比第二个要大50%。

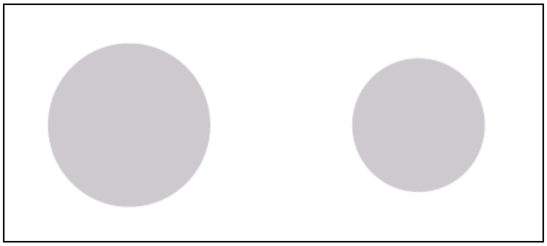


图6-13　通过面积对比的两个气泡

而在图6-14中则是根据半径的长短来表现两个气泡的大小关系。

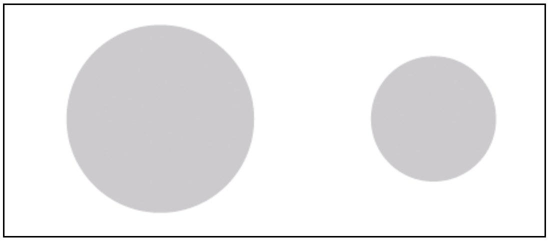


图6-14　通过半径对比的两个气泡

第一个圆的半径比第二个圆的半径要长50%，这使得第一个圆的面积超过了第二个图的2倍还要多。虽然看上去这还不算什么大问题（现在只有两个数据点，比较起来还算容易），但在尝试表现更多数据时，这一问题就会很容易造成困扰。

创建气泡图

让我们看一下将要实现的最终图表效果，如图6-15所示。它依然是有关美国各州谋杀案和入室盗窃案的数据，但加入了各州的人口数量作为第三维度。是否人数越多的州，犯罪率也会越高？事实上并没有这么单纯（和大多数情况一样）。像加利福尼亚、佛罗里达和得克萨斯这些大州确

实很接近象限的右上部分，而纽约州和宾夕法尼亚州的入室盗窃率却相对较低。与之类似，路易斯安那和马里兰的人口较少，但却位于图表的最右边。

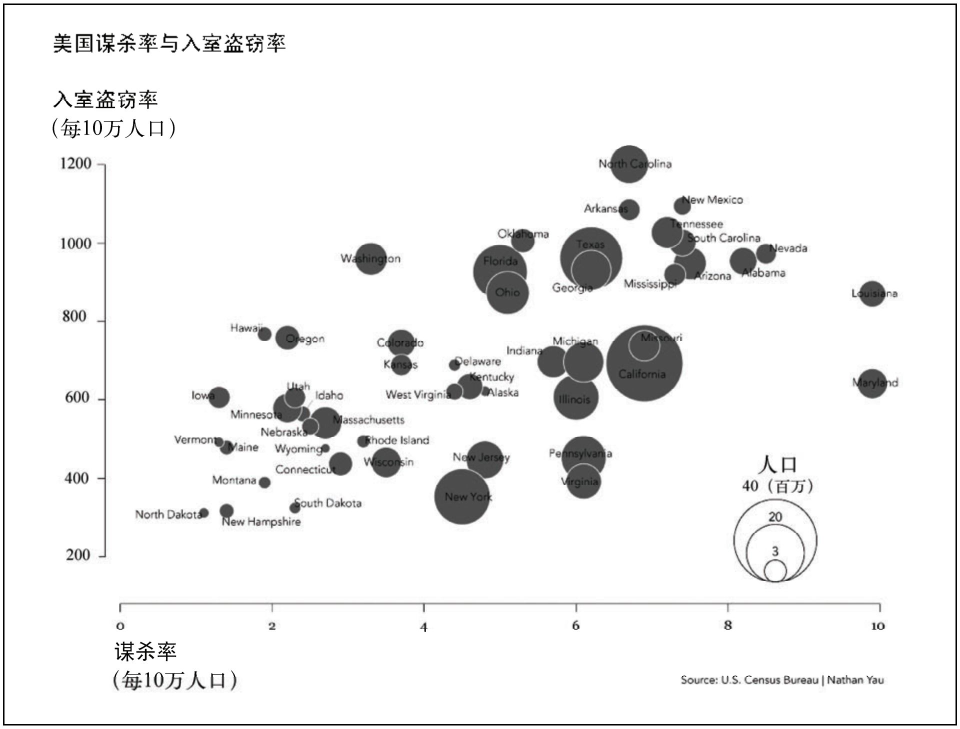


图6-15　显示美国犯罪率的气泡图

首先在R中用`read.csv()`载入数据。这次的数据和我们之前所使用的基本上相同，只不过加入了一列人口数量，而华盛顿特区被移除了。另外它不是通过逗号分隔，而是制表符分隔。不过这没什么大不了的，只需在函数内稍微改动一下`sep`参数即可。

```
crime <-
    read.csv("http://datasets.flowingdata.com/crimeRatesByState2005.tsv",
    header=TRUE, sep="\t")
```

之后就可以直接用`symbols()`函数来绘制气泡了。*x*轴是谋杀率，*y*轴是入室盗窃率，而气泡的半径是根据人口数量计算得出的。结果如图6-16所示。想尝试一下`symbols()`的其他功能吗？在后文中马上就会看到。

```
symbols(crime$murder, crime$burglary, circles=crime$population)
```

搞定，对吧？错，这是用来考验你的。这些圆的大小是根据人口比例换算成半径来定义的，而我们需要的是根据面积来定义。因为如果按半径来指定的话，相互之间的比例就会紊乱。加利福尼亚州的人口（由中间那个巨大的圆形代表）真的比其他州要多出那么多吗？

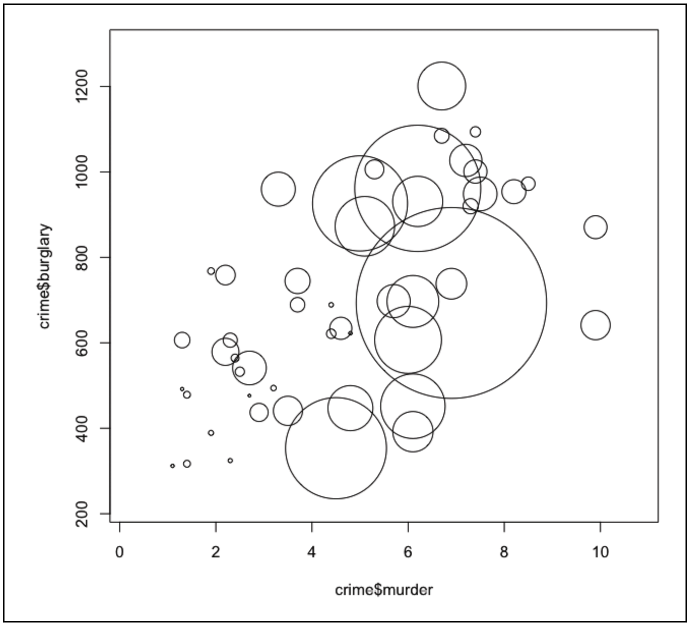


图6-16　默认生成的气泡图

要想正确地定义大小，先来看一下圆面积的计算公式。

$$圆的面积 = \pi r^2$$

代表人口数量的是每一个气泡的面积。我们需要弄明白怎样通过半径来定义大小，所以把半径移到等号左侧可以发现，它应该是和面积的平方根成正比例的。

$$r = \sqrt{圆的面积/\pi}$$

你可以直接去掉π，因为它只是一个常量，不过为了清楚起见，还是把它留在那里。现在就不能只通过`crime$population`来指定圆的半径了，而应该计算出它的平方根，然后插入到`symbols()`里面。

```
radius <- sqrt( crime$population/ pi )
symbols(crime$murder, crime$burglary, circles=radius)
```

第一行代码创建了一个平方根取值的新向量，并存储到`radius`中。图6-17显示了正确大小的气泡图，但显得比较杂乱，因为那些人口少于加利福尼亚的州的气泡都变大了。

我们需要把所有的圆都按比例缩小才能看清楚图表的内容。`symbols()`中的`inches`参数可以设定最大圆形的大小，单位是英寸。默认的大小是1英寸，所以在图6-17中加利福尼亚州气泡的大小就是1英寸，其他的气泡都据此按比例缩小。我们可以缩小最大的气泡（比如说0.35英寸），但比例保持不变。此外还可以通过`fg`和`bg`参数来改变每个圆的边框和填充色，并且为坐标轴加上你自己的标签。图6-18显示了最终的输出结果。

```
symbols(crime$murder, crime$burglary, circles=radius, inches=0.35,
    fg="white", bg="red", xlab="Murder Rate", ylab="Burglary Rate")
```

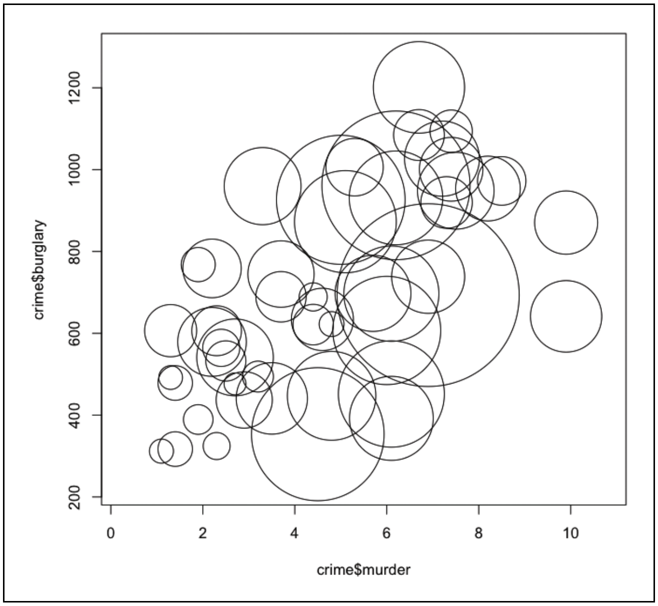


图6-17　默认生成的正确尺寸的气泡图

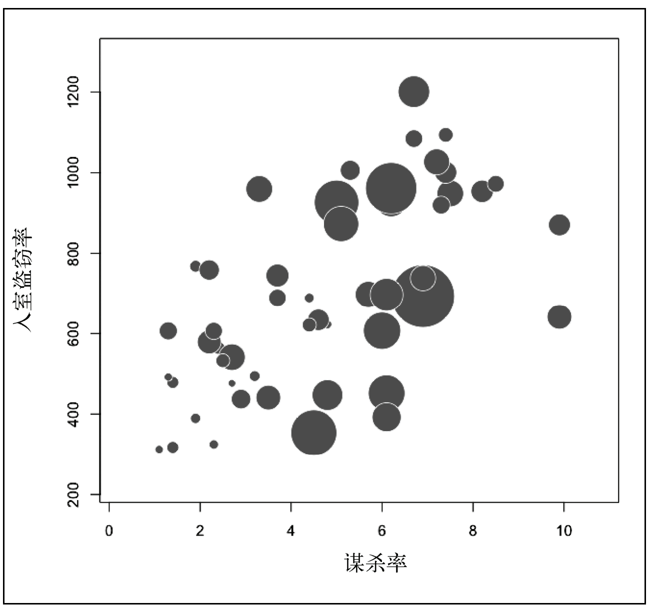


图6-18　气泡缩小后的气泡图（另见彩插图6-18）

现在的效果有点感觉了。

你也可以通过`symbols()`用其他形状来创建图表，例如正方形、矩形、量度表、箱形图和星形图。它们的参数和使用圆形时稍有不同。比如说正方形，就是用它的边长来定义大小。不过和气泡一样，我们需要用面积来表现各个正方形之间的关系，换句话说，需要通过面积的平方根来指定正方形的边长。

图6-19显示了使用正方形后图表的样子。代码如下：

```
symbols(crime$murder, crime$burglary,
    squares=sqrt(crime$population), inches=0.5)
```

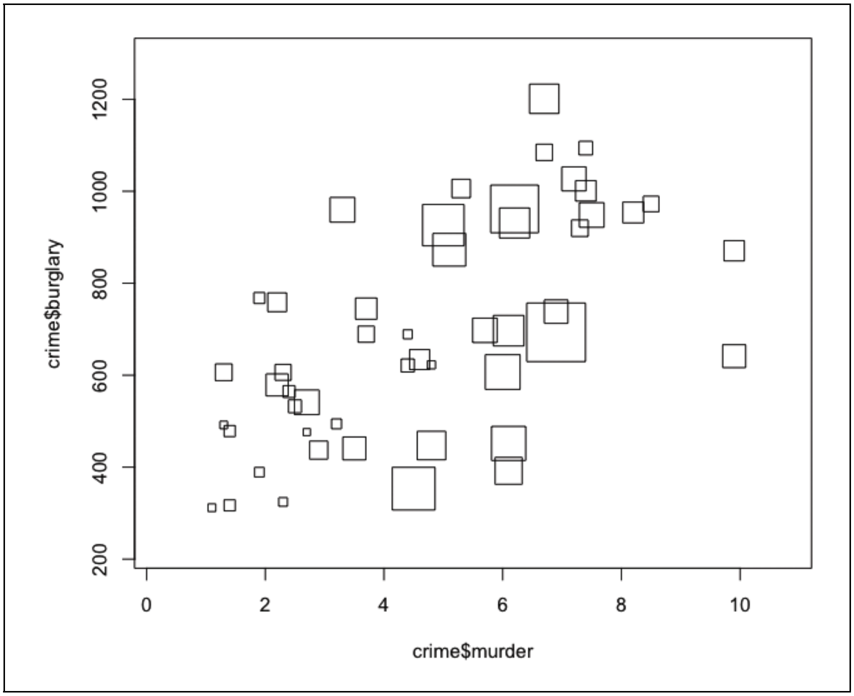


图6-19 用正方形代替圆形

现在还是让我们回到圆形。图6-18中的图表已经有了分布的感觉，但是读者还是不知道哪个圆代表哪个州。所以我们还需要添加标签。这会用到`text()`函数，它的参数包括*x*轴坐标、*y*轴坐标和具体显示的文字内容，这些我们手上都有。和代表各州的各个气泡一样，各州名的*x*轴坐标来自于各州的谋杀率，*y*轴坐标来自于各州的入室盗窃率。具体显示的文字则是各州的名称，来自于我们的原始数据块中的第一列。

```
text(crime$murder, crime$burglary, crime$state, cex=0.5)
```

其中`cex`参数控制的是文本的大小，默认值为1。如果取值大于1，标签就会更大，反之则会更小。标签默认是以*x*轴和*y*轴坐标居中对齐的，如图6-20所示。

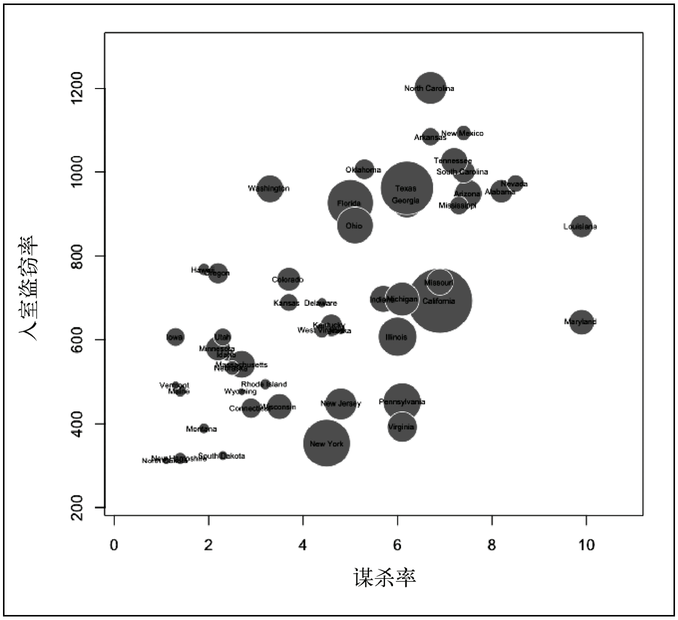


图6-20　带有标签的气泡图

到了这一步，已经很接近图6-15所示的最终效果了。将R中生成的图表另存为PDF格式，然后用你自己擅长的绘图软件进行调整，把它改成你希望的样子。你可以对图表进行简化，比如让坐标轴变得更细，并且删除四周的边框。你也可以调整某些标签的位置，尤其是位于左下角的那几个州，以便读者辨识出它们的名字。此外需要把乔治亚州的气泡挪到上方，因为它被面积更大的得克萨斯州给挡住了。

这就是气泡图。在R中输入`?symbols`可以了解更多绘图选项。不妨大胆一点，多试试。

6.3　分布

大家也许听说过平均数（mean）、中位数（median）和众数（mode）。美国一般会在高中教这些东西，不过我觉得这也有点太晚了。平均数指的是将所有数据点相加然后除以数据点的个数。中位数指的是将所有数据按从小到大依次排列，然后找到位于最中间的那个数据点。众数则是指在所有数据点中出现频率最高的那个数字。这些数字都有其价值，也很容易得出，但是它们仅仅描述了数据大概的分布情况，无法展现出故事的全貌。而用可视化的方式来表现所有的内容，其中的分布自然也就一目了然了。

图表的重心如果向左偏斜，则表明大部分数据都聚集在整个取值域较低的一端。向右偏斜则含义相反。平坦的线意味着均匀分布，而经典的钟形曲线则表明大部分数据聚集在平均值附近，同时向两端逐渐减少。

接下来会先向大家介绍一种经典的图表类型，让你对分布有一个更直观的认识，之后我们将

讨论更加实用的直方图（histogram）和密度图（density plot）。

6.3.1 老式的分布图表

在20世纪70年代，计算机还远未普及，绝大部分的数据图表都是手绘的。著名的统计学家John Tukey就在他的著作《探索性数据分析理论》（*Exploratory Data Analysis*）中给出了一些建议，基本上都是关于如何用钢笔和铅笔来区分线条和色彩的明暗度。你也可以用各种花纹作为填充，从而对变量进行区分。

茎叶图（stem-and-leaf plot，或者stemplot）就是在这种环境下被创造出来的。绘制者所要做的就是按照某种顺序来填写数字，最终得到的结果可以让读者对数据的分布有个大致的了解。这一方法在20世纪80年代非常流行（当时在分析中运用统计图表的势头开始增强），因为它制作简单，而且就算是用打字机也能营造出图形感。

虽然说在今天我们有其他更简单、更快捷的方式可以观察分布，但回顾一下这种方式仍然很有价值，因为在制作茎叶图时用到的原则也同样适用于直方图。

创建茎叶图

如果你想回归自然，可以用钢笔在纸上绘制茎叶图。不过用R会快得多。图6-21显示了2008年全球出生率的茎叶图，数据来自于世界银行。

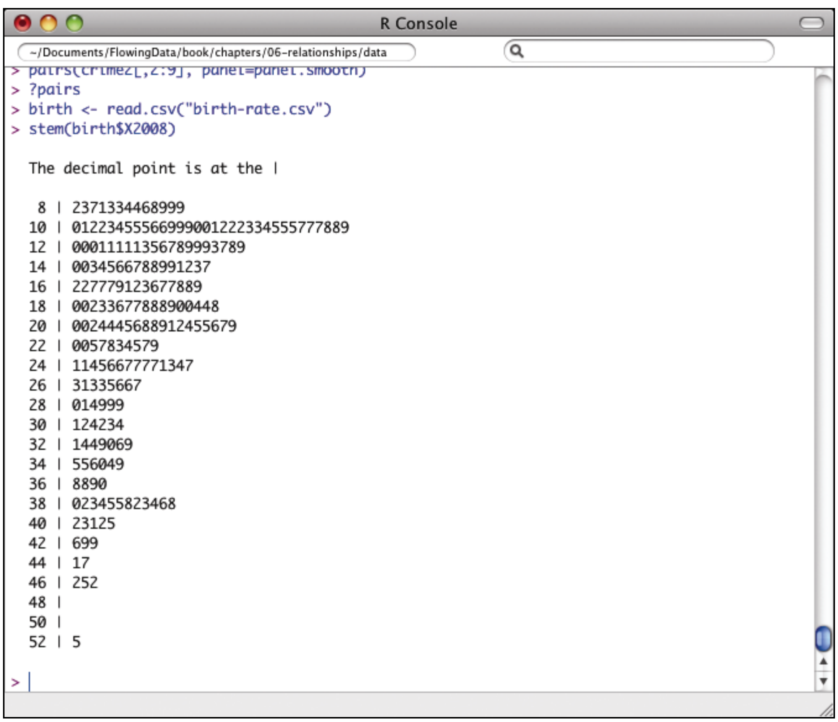


图6-21 显示全球出生率的茎叶图

如你所见，它非常基础。基础性的数字位于左侧（茎），与其相关的数字依次排列在右侧（叶）。在本例中，竖线（|）表示小数点，左侧代表整数，右侧代表各数据的小数位第一个数字。所以在2008年，世界各国每1千人口中的出生率最常见的区间是10到12之间。不难看出，有一个国家（尼日尔）的出生率最高，在52到54之间。

以下是手绘茎叶图的方法。从8开始写下数字，以52结束，其中以2为间隔，按从上到下排列。在这一列数字的右侧画一条竖线，然后逐行检查2008年的各国数据，并相应在竖线的右侧添加各数据小数点后的第一个数字（如果有多个数字，则四舍五入）。比如说，某个国家的出生率是8.2，那么就在数字8的右侧添加一个数字2。如果有一个国家的出生率是9.9，它依然也会归于数字8那一行，我们就应该在后面接着添加一个数字9。

很明显，如果数据很多，这个过程就会相当吃力。因此我们来看看怎样在R中绘制茎叶图。在载入数据之后，只需调用`stem()`函数即可。

```
birth <- read.csv("http://datasets.flowingdata.com/birth-rate.csv")
stem(birth$X2008)
```

这就完成了。如果你想让它更美观一点，比如说像图6-22那样，那么可以复制R中的文本，然后粘贴到其他地方调整样式。不过这种方法已经过时了，用直方图可能会好很多。其实直方图基本上就是更加图形化的茎叶图。

```
 8 | 2371334468999
10 | 01223455566999001222334555777889
12 | 00011111356789993789
14 | 0034566788991237
16 | 227779123677889
18 | 00233677888900448
20 | 0024445688912455679
22 | 0057834579
24 | 11456677771347
26 | 31335667
28 | 014999
30 | 124234
32 | 1449069
34 | 556049
36 | 8890
38 | 023455823468
40 | 23125
42 | 699
44 | 17
46 | 252
48 |
50 |
52 | 5
```

图6-22　稍加改进后的茎叶图

6.3.2 有关分布的柱形

观察图6-22中的茎叶图，我们可以辨识出每个具体取值域各自的出现频率。某个出生率取值域内的国家越多，绘制出的数字就越多，因此所处的那一行也会越长。现在逆时针翻转茎叶图，让它的行变成列，可以发现越高的列表示该取值域内的国家越多。现在把每一列的数字改成简单的柱形（或者说矩形），我们就得到了一个直方图，如图6-23所示。

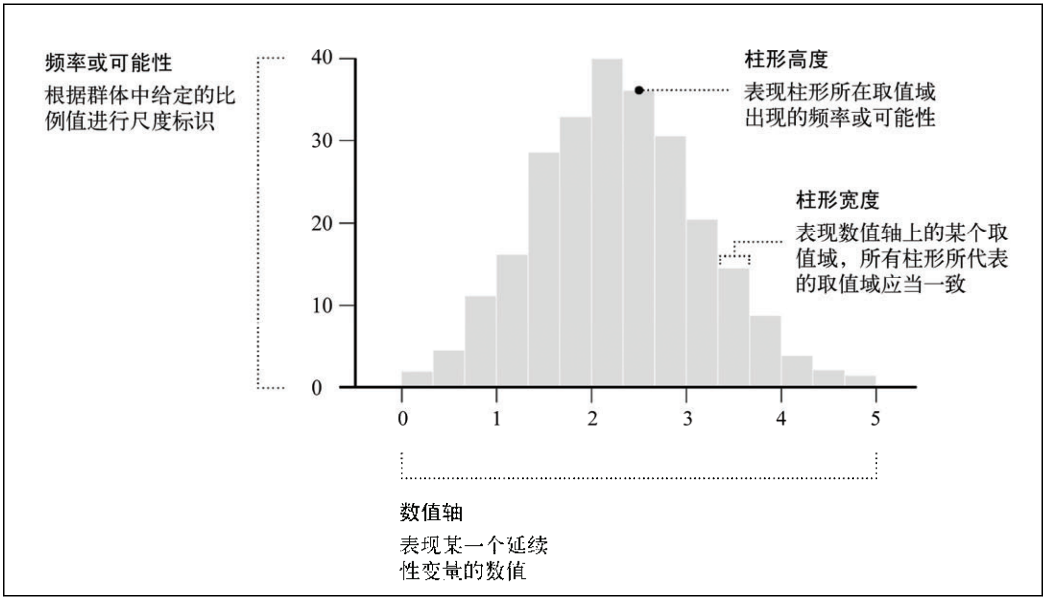


图6-23 直方图的基本框架

柱形的高度表示频率，而柱形宽度没有具体意义。直方图的水平轴和垂直轴都是连续的，而一般柱形图的水平轴上各个数值则是相互分离的。在使用柱形图时，一般会通过水平轴表现各种类别，而且各个柱形之间通常会留有间隙。

有许多人不像我们这样长期与数据图表打交道，他们往往会误认为水平轴只能是时间。水平轴可以是时间，但绝不仅限于此。在考虑我们的受众群体时这一点非常重要。如果你的图表主要呈现给普通读者，那么就需要解释图表的阅读方法，以及他们需要注意的地方。另外要记住，仍然有很多人并不熟悉分布的概念，所以在设计图表时一定要清晰，并且尽量设身处地地帮助读者。

创建直方图

和茎叶图一样，用R创建直方图也非常容易。利用`hist()`函数再次绘制全球出生率的分布图，结果如图6-24所示。注意到它的形状与图6-22的茎叶图之间的相似之处了吗？

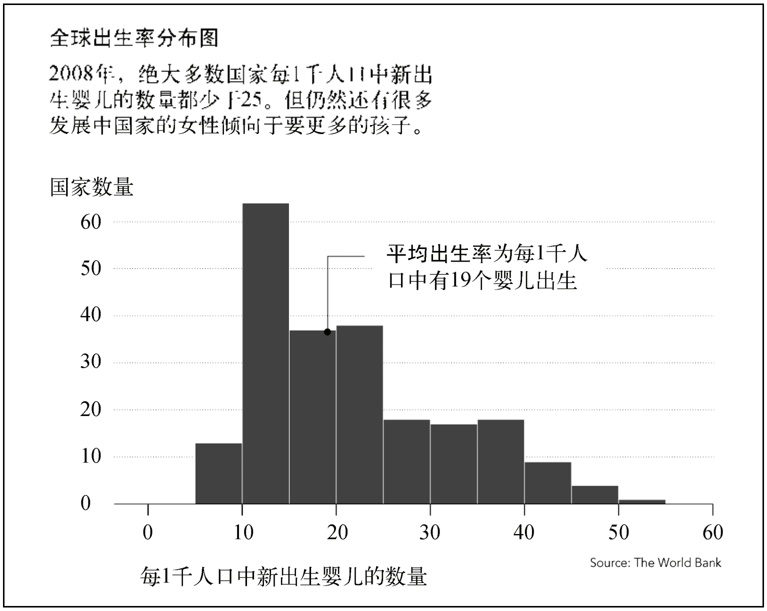


图6-24　全球出生率分布图

假设你在之前的例子中已经载入了数据，那么现在可以直接用`hist()`函数来处理2008年的数据了。

```
hist(birth$X2008)
```

图6-25显示了默认生成的直方图。

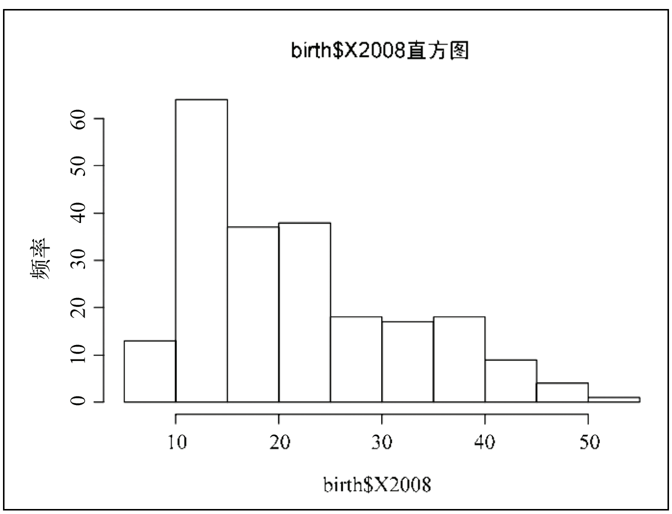


图6-25　默认生成的直方图

在默认生成的直方图中有10个柱形（或者说取值域），但我们可以通过breaks参数随心所欲地修改。比如说可以设置得到如图6-26所示的更少、更宽的柱形。它只分成了5段。

```
hist(birth$X2008, breaks=5)
```

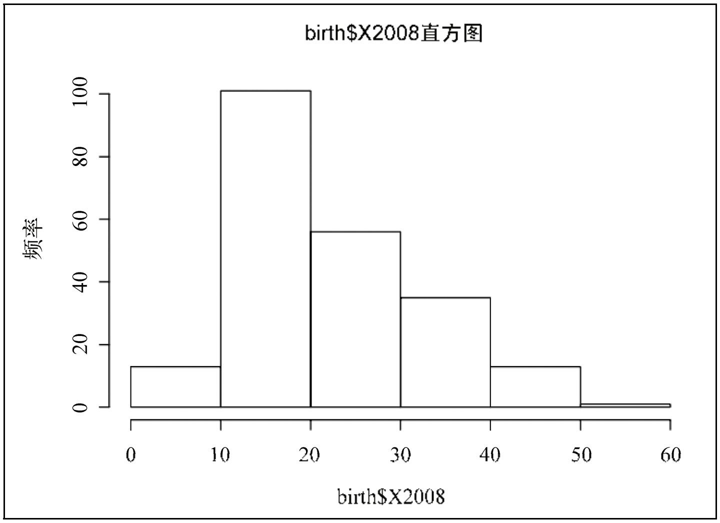


图6-26　分成5段的直方图

我们也可以反其道而行之，让直方图显示更多较窄的柱形，比如说20个（参见图6-27）。

```
hist(birth$X2008, breaks=20)
```

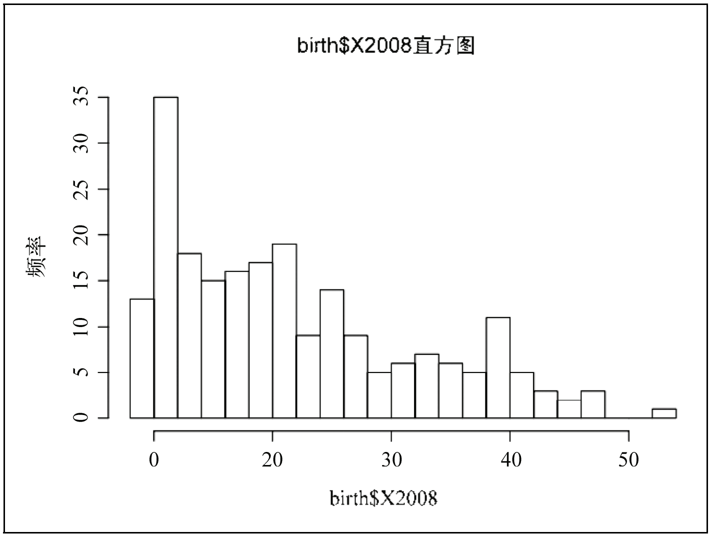


图6-27　分成20段的直方图

你应该根据所要可视化的数据特点来决定分段的数量。如果大部分数据都聚集在某个取值域内，那么就应该采用较多的分段，以便观察其中的细节变化，而不是只生成一个很高的柱形。但如果没有那么多数据，或者数字的分布比较平均，那么较粗的柱形就会更加合适。好消息是如果想试验各种不同数量的柱形，在操作上没有任何难度。

提示　默认的直方图分段数量并不一定就是最佳选择。不断调整参数，选择最适合数据特点的分段数量。

对于出生率数据来说，默认的分段数量还算合理。我们可以看到有一些国家每1千人口中新生婴儿的数量在10以下，但绝大多数国家的这一数字都位于10到25之间。还有不少国家超过了25，但总体数量要少于25以下的国家。

现在可以将图表另存为PDF格式，并在Illustrator中继续编辑。大部分操作都和我们在第4章中针对柱形图所做的类似，不过也可以加入一些直方图特定的内容，从而提高它的可读性。另外最好向读者作一些解释说明。

提示　利用R的`summary()`函数，我们可以轻松获取平均数、中位数、最大值、四分位数等关键数字。

在图6-24的最终图表中，你可以看到分布的一些关键点，比如平均值、最大值和最小值等。当然，文字说明也是一个提供解释说明的好机会。此外加入一点颜色会好很多，这样直方图看起来就不那么像线框了。

6.3.3　延续性的密度

虽然直方图的数值轴是延续性的，但整个分布依然被分成了数个柱形。每一个柱形代表的都是一些条目（在上例中也就是国家）的集合。那么在每个柱形内部的变化又是怎样的呢？在茎叶图中，我们确实可以看到每个国家的具体数字，但要想就此估计各国间差距的大小，依然不够直观。其实和第4章中用LOESS曲线来观察趋势一样，我们也可以用密度图来对分布的细节变化进行可视化。

图6-28显示了用曲线代替柱形图的效果。曲线以下的总面积等于1，垂直轴代表的是可能性，或者说样本群体中某个值所占的比例。

创建密度图

让我们回到出生率的数据，要生成密度图还需要一个步骤，不过这个步骤并不复杂。我们需要用`density()`函数来估算曲线中的各个数据点，而调用的数据中不能有任何缺失值。在2008年的数据中，有15行都缺失了具体数值（15个国家没有数据）。

在R中，这些缺失值都被标记为`NA`。幸而我们很容易就能把这些值过滤出去。

```
birth2008 <- birth$X2008[!is.na(birth$X2008)]
```

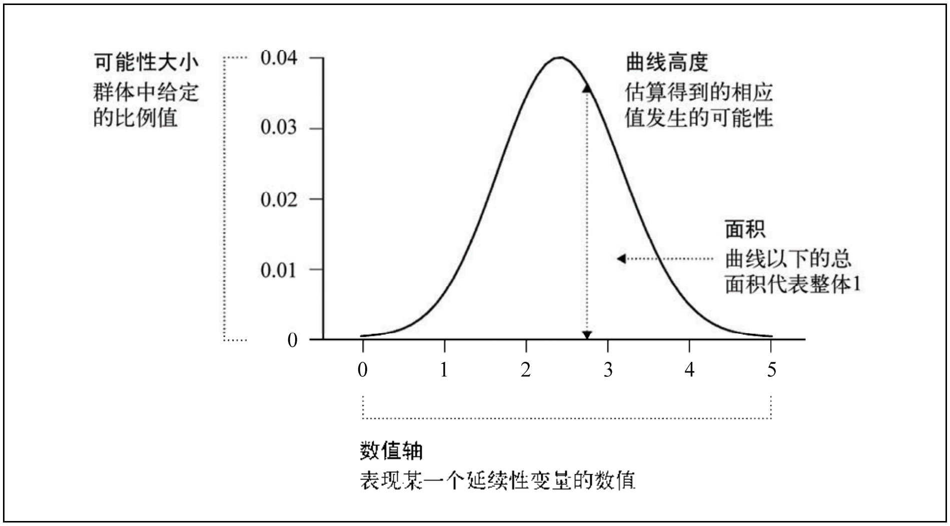


图6-28 密度图的基本框架

说明 在本例中，为简化起见，我们把缺失值都直接移除了。在你自己探索和可视化数据时，最好多注意一下这些缺失值。它们为什么会缺失？是应该把它们归零呢，还是直接移除？

这行代码会从出生率数据中检索2008年的那一列，然后只读取那些不含有缺失值的行，并存储到变量`birth2008`中。用技术性语言来描述就是：`is.na()`会检测`birth$X2008`这个向量中的每一个条目，然后通过布尔运算判断各个条目的值是`true`还是`false`，最后返回一个长度相同的向量。当我们将这个布尔运算得到的向量传递到原向量的索引中时，就只会返回其中那些值为`true`的条目了。如果你不太理解这些也没关系，我们无需了解所有的技术细节也一样能够得到结果。不过，如果你打算定义自己的函数，以上内容可能会帮助你了解代码的原理。

现在我们用`birth2008`储存了“干净”的出生率数据，所以可以将它传递给`density()`函数以求出曲线，并将结果存储到`d2008`中。

```
d2008 <- density(birth2008)
```

这行代码可以让我们得到曲线的各个*x*轴和*y*轴坐标。我们甚至还可以把这些坐标存储为文本文件，这一点很酷，因为有了这个文本文件，就可以导入到其他的图表绘制软件中去。在R的控制台输入`d2008`可以看到变量中存储的内容，如下所示：

```
Call:
    density.default(x = birth2008)

Data: birth2008 (219 obs.);    Bandwidth 'bw' = 3.168
```

6

```
       x                 y
 Min.   :-1.299   Min.   :6.479e-06
 1st Qu.:14.786   1st Qu.:1.433e-03
 Median :30.870   Median :1.466e-02
 Mean   :30.870   Mean   :1.553e-02
 3rd Qu.:46.954   3rd Qu.:2.646e-02
 Max.   :63.039   Max.   :4.408e-02
```

我们关心的主要是其中的x和y。以上显示的是各坐标的细目，如果你想获得所有的坐标，可以按以下代码输入：

```
d2008$x
d2008$y
```

然后试着用write.table()将它们保存到文本文件中。该函数中的参数可以指定你希望保存的数据、文件名以及希望使用的分隔符（如逗号或制表符），以及其他一些内容。要想把该数据存为基本的以制表符分隔的文本文件，可以按以下代码输入：

```
d2008frame <- data.frame(d2008$x, d2008$y)
write.table(d2008frame, "birthdensity.txt", sep="\t")
```

说明　`write.table()`函数会在你当前的工作路径中生成一个新文件。可以通过主菜单或者`setwd()`来改变当前的工作路径。

现在你的工作路径中应该出现了一个birthdensity.txt文件。如果你不希望每行都列出序号，或者不希望用制表符而是用逗号作为分隔符，也很容易实现：

```
write.table(d2008frame, "birthdensity.txt", sep=",", row.names=FALSE)
```

现在你可以把这些数据导入到Excel、Tableau、Protovis，或者其他任何能识别分隔符文本的软件中去了。几乎所有软件都支持这种格式。

提示　如果你想用R以外的软件来绘图，但又不希望放弃R的计算功能，可以在任何时候通过`write.table()`函数来保存当前R中的结果。

让我们回到之前的密度图。我们已经得到了密度图的坐标，现在只需用`plot()`函数将它们呈现为图表格式即可。结果如图6-29所示。

```
plot(d2008)
```

如果你愿意，也可以通过`plot()`加上`polygon()`函数创建带有填充色的密度图，效果如图6-30所示。前者用于设置坐标轴，但将`type`（类型）设为`"n"`表示暂不绘制。接着用后者来绘制形状，并且将填充色设为暗红色、边框设为浅灰色。

```
plot(d2008, type="n")
polygon(d2008, col="#821122", border="#cccccc")
```

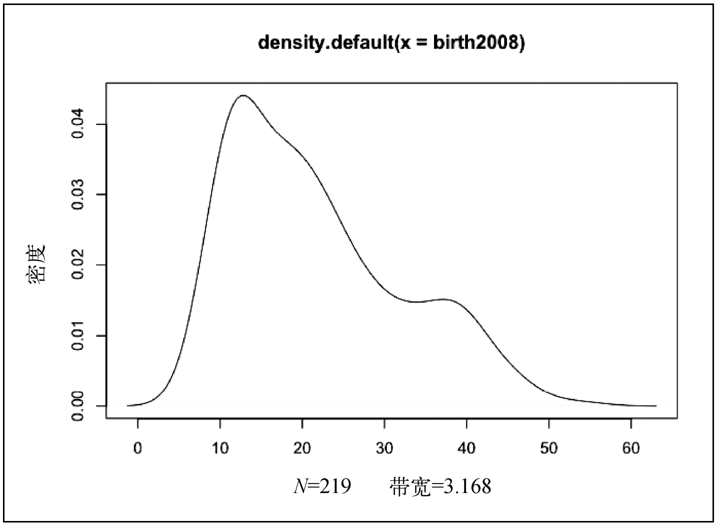


图6-29　出生率的密度图

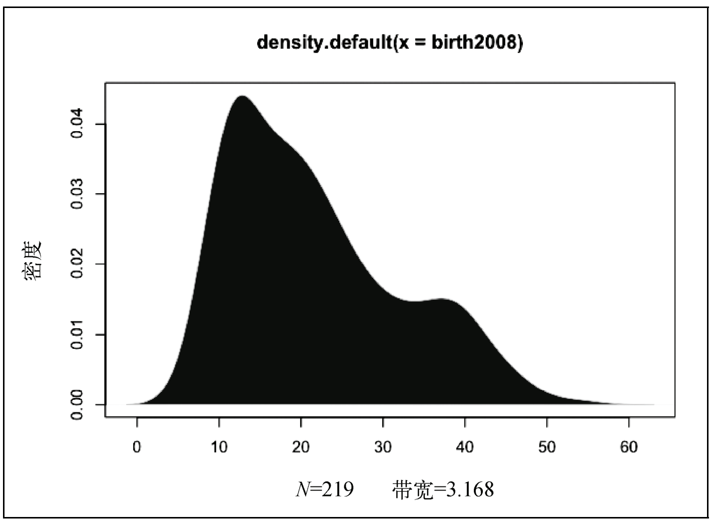


图6-30　填充后的密度图

现在还可以继续深入一步。可以将直方图和密度图绘制在一起，通过柱形来表现实际频率的分布、通过曲线来表现计算得出的比例，如图6-31所示。这里用到的是`histogram()`（来自于lattice工具包）和`lines()`这两个函数。前者创建一个新的图表，而后者在已有的图表上加上线条。

```
library(lattice)
histogram(birth$X2008, breaks=10)
lines(d2008)
```

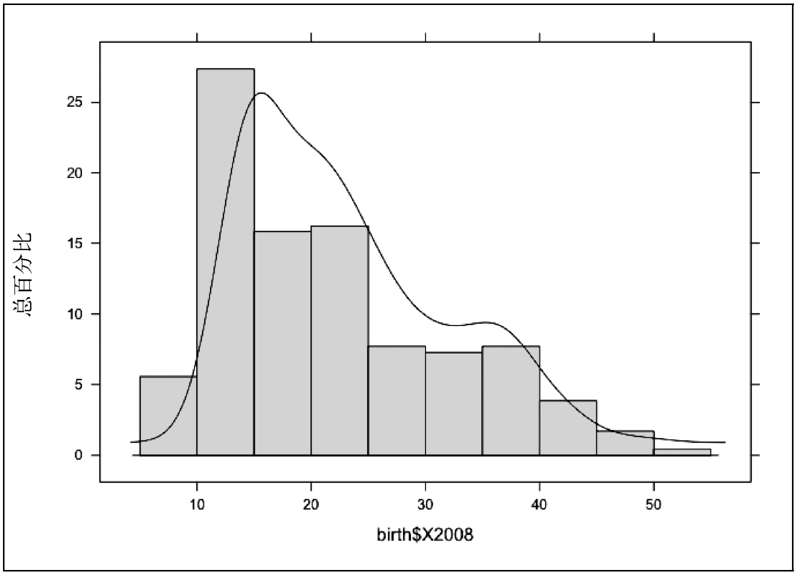


图6-31　直方图和密度图的混合

我们能做的有很多，还可以加入各种变化，但其中涉及的数学和几何学知识其实和老式的茎叶图是一样的，无非是计算、聚合然后分组。我们可以根据数据对象的特点来决定选择何种变体。图6-32显示了经过修饰后的图表。我弱化了坐标轴，重新添加了标签，并标出了平均值。表现密度的垂直轴在本图中的作用不大，不过出于完整性的考虑我依然还是保留了它。

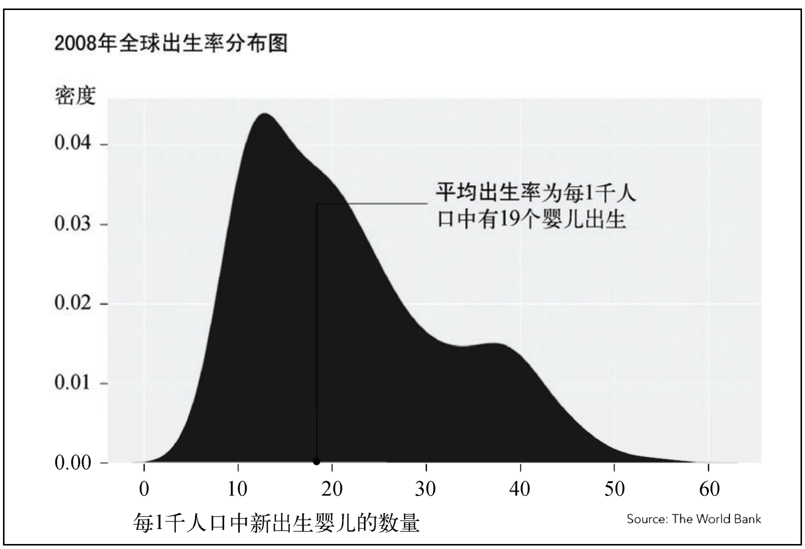


图6-32　2008年全球出生率的密度图

6.4 对照和比较

通常来说，将多个分布同时进行比较会带来更大帮助，而不是只考虑平均数、中位数和众数。毕竟这些摘要性质的统计只是对大局的一种“描述符”，它们讲述的只是故事的片段。

比如说，我可以告诉你2008年的全球平均出生率是每1千人口中有19.98个婴儿出生，而在1960年的平均数则是32.87。所以2008年的出生率要比1960年低39%。但这些只能告诉你整个分布的总体发生了什么。是否大多数国家的出生率其实并无显著变化，只是有少数国家在1960年的出生率过高，从而拉动了平均值？过去这几十年来出生率间的差距到底是在持续增加还是持续减少？

我们可以通过多种途径来进行比较。我们可以摒弃可视化、只靠分析来得出结果（我读研时花了一整年时间学习各种统计方法，而它们也只是冰山一角）。或者我们也可以利用可视化来达到同样的效果。这种方法得到的结果虽然不如周密的统计分析那么精确，但它们足以帮助我们在任何方面作出明智的决策。很明显，更多人都会选择可视化途径，这也是我写这本书的目的，不是吗？

多个分布图表

到目前为止我们只专注于一个分布情况，也就是2008年的出生率。但如果查看一下数据源文件或者R中的数据帧，我们就会发现其中提供了1960—2008年每年的出生率数据。唔，当然了，无论你看不看，它们都在那里。正如我之前所提到的，全球出生率呈现显著的下降趋势，但整个分布到底是如何变化的呢？

现在让我们直接为每一年都创建一个直方图，然后把它们以矩阵的形式进行组织。这种设计思路和本章一开始的散点图矩阵非常相似。

1. 创建直方图矩阵

R中的lattice工具包让我们只需一行代码就能创建出一整套直方图，但有一个小问题：我们必须以这个函数接受的格式来向它传递数据。以下是R在一开始载入的文本文件的片段。

```
Country,1960,1961,1962,1963...
Aruba,36.4,35.179,33.863,32.459...
Afghanistan,52.201,52.206,52.208,52.204...
...
```

可以看出，每一个国家都有一行数据。第一列是国家的名称，然后每一年都占有一列，因此算上标头共有50列234行数据。但我们需要这些数据按两列来呈现：一列是年份，另一列是出生率。这里并不需要国家的名称，因此数据的头几行看起来应该是这个样子：

```
year,rate
1960,36.4
1961,35.179
1962,33.863
1963,32.459
1964,30.994
1965,29.513
...
```

比较一下这两个片段，不难发现在第二个片段中的数值对应的是前一个片段中的国家阿鲁巴（Aruba，加勒比海岛国）。所以每一个国家每一年的出生率都应占据一行，这样就会需要9870行数据（算上标头）。

那么怎样才能让数据变成我们想要的格式呢？还记得在第2章中用过的Python吗？我们在Python里载入CSV文件，然后循环处理其中的每一行，输出为我们想要的数据格式。现在也可以这么干。用你喜欢的文本编辑器新建一个文本文件并存为transform-birth-rate.py。请确保它的存放路径与birth-rate.csv相同。然后输入以下代码：

```
import csv

reader = csv.reader(open('birth-rate.csv', 'r'), delimiter=",")

rows_so_far = 0
print 'year,rate'
for row in reader:
    if rows_so_far == 0:
        header = row
        rows_so_far += 1
    else:

        for i in range(len(row)):
            if i > 0 and row[i]:
                print header[i] + ',' + row[i]

        rows_so_far += 1
```

以上代码看上去应该很眼熟，不过现在让我们拆开来分析。我们导入了csv工具包，接着载入birth-rate.csv。然后输出标头，循环处理每一行每一列，从而让代码把数据输出为我们想要的格式。在控制台中运行以上脚本，并将输出保存为一个名为birth-rate-yearly.csv的新CSV文件。

```
python transform-birth-rate.py > birth-rate-yearly.csv
```

提示　如果你希望将全部代码都写在R里，可以试试Hadley Wickham的reshape工具包。它能帮助你把数据帧改为需要的格式。

很好。现在用`histogram()`来创建矩阵：让我们回到R，用`read.csv()`载入新的数据文件。如果你刚才省略了改变数据格式那一步，我这里提供了一份修改后的数据文件，你可以直接通过URL来载入它。

```
birth_yearly <-
    read.csv("http://datasets.flowingdata.com/birth-rate-yearly.csv")
```

现在把数据插入到`histogram()`里面，生成一个10 × 5的矩阵，显示按年份划分的出生率。输出结果应如图6-33所示。

```
histogram(~ rate | year, data=birth_yearly, layout=c(10,5))
```

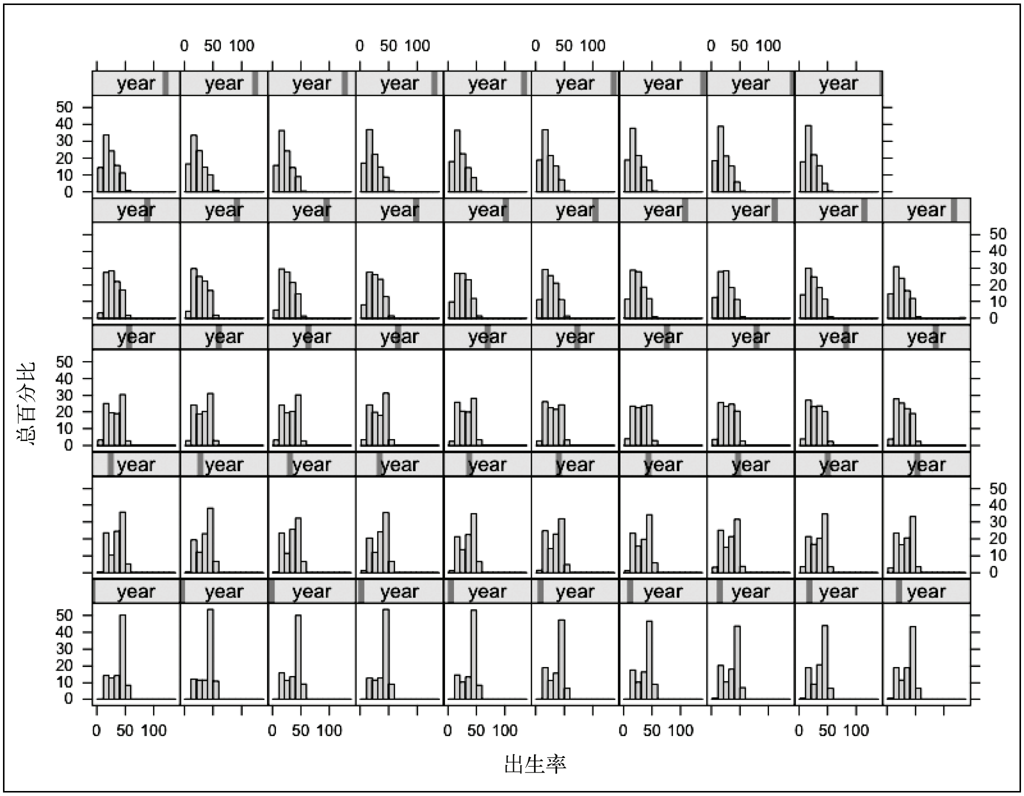


图6-33 默认生成的直方图矩阵

效果不赖，但还有改善的余地。首先，有一个异常值出现在右侧很远的位置，导致其他所有柱形都被挤到了左侧。其次，矩阵的每一个单元中都有根橙色的竖线，随着年份的增加它会从左往右移动，但如果明确标出年份数字，反而会更加便于理解。最后一点（由于没有年份标签所以很难看出），这些直方图的顺序并不太合适。第一个年份，也就是1960年，出现在左下角，1969年在右下角。1960年那个单元的上方是1970年。也就是说，年代顺序是从下往上、从左至右排列的。这让人感觉很别扭。

要想找出那个异常值，让我们再次对`birth_yearly`使用`summary()`函数。

```
     year            rate
Min.   :1960   Min.   :  6.90
1st Qu.:1973   1st Qu.: 18.06
Median :1986   Median : 29.61
Mean   :1985   Mean   : 29.94
3rd Qu.:1997   3rd Qu.: 41.91
Max.   :2008   Max.   :132.00
```

最大值是132。这看起来有些离谱，因为出生率几乎没有接近100的。是哪里出了问题？我们发现帕劳（西太平洋岛国）在1999年的出生率被记录为132。这很可能是一个笔误，因为帕劳的出生率在1999年前后都从未超过20。很可能应该是13.2，但我们也不能妄下结论，最好多加研究。就目前来说，让我们先移除这个数值。

```
birth_yearly.new <- birth_yearly[birth_yearly$rate < 132,]
```

再来看看年份标签。如果标签的值是以数字存储的，那么lattice函数会自动用一根橙色竖线来指示数值。然而，如果标签处是字符，那么函数就会使用字符串，所以现在让我们来调整一下：

```
birth_yearly.new$year <- as.character(birth_yearly.new$year)
```

我们还需要调整顺序。首先还是得创建直方图矩阵，然后将其存储为一个变量。

```
h <- histogram(~ rate | year, data=birth_yearly.new, layout=c(10,5))
```

然后用`update()`函数来改变直方图的顺序。

```
update(h, index.cond=list(c(41:50, 31:40, 21:30, 11:20, 1:10)))
```

这样就调转了所有行的顺序。结果如图6-34所示，我们得到了一个标签清晰的直方图矩阵，而且由于移除了错误值，分布呈现得更加明确。此外，直方图的排列更具有逻辑性，读者可以从左至右、从上到下地阅读。视线从每一行中某个单元直接往下移动，就能看到分布在10年后的变化。

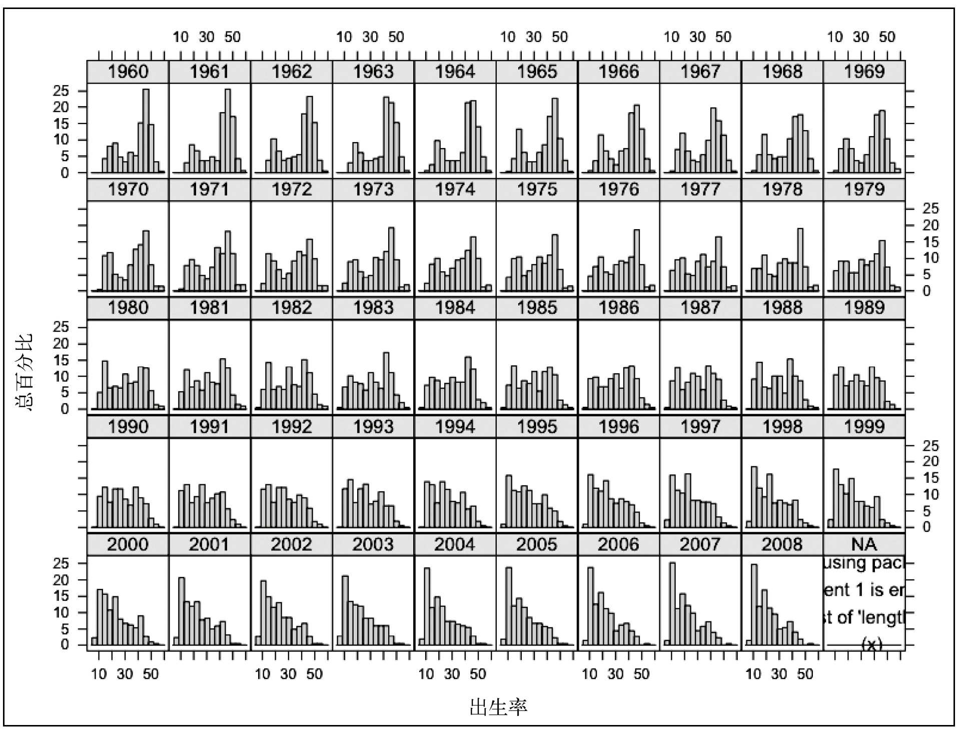


图6-34　修改后的直方图矩阵

现在的整个布局已经很不错了。如果你希望在Illustrator里继续美化图表，可以缩小标签的字体、调整边框和填充色，以及一些整体上的清理，结果如图6-35所示。这样一来可读性会得到进一步增强。要想让图表的意思更加明确，并且讲述故事的来龙去脉，你还可以添加适当的文字说明、数据来源，并且指出全球出生率的分布是如何逐渐往左（低出生率）偏移的。对于单独的图表来说这可能有点复杂，因此你需要提供充分的上下文背景，以便于读者完全理解数据的意义。

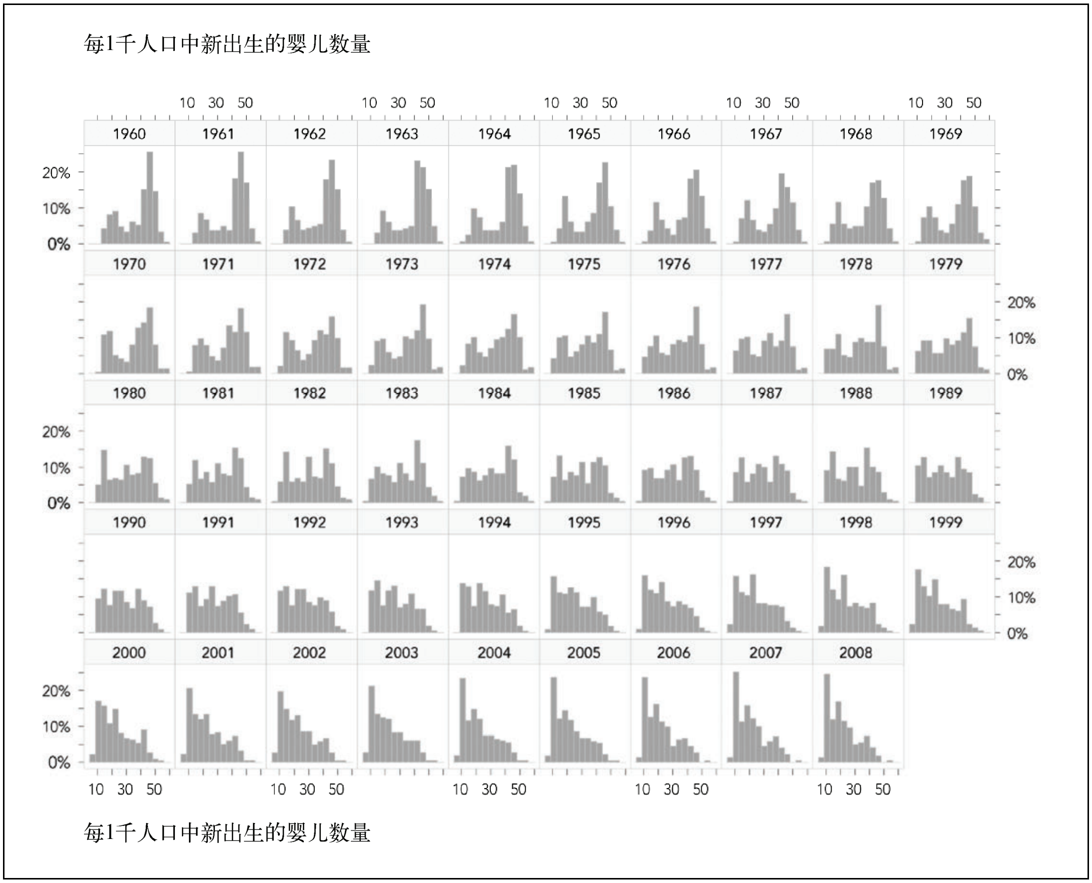


图6-35　在Illustrator 中完善后的直方图矩阵

解决问题并不是只有这一种途径。我们还可以用Processing、Protovis、PHP或者任何能绘制柱形的工具来创建同样的直方图矩阵。甚至在R里也有多种方法可以绘制和上图一样的矩阵。比如说，我为FlowingData绘制了一幅有关2002—2009年家庭电视尺寸的分布变化图，如图6-36所示。

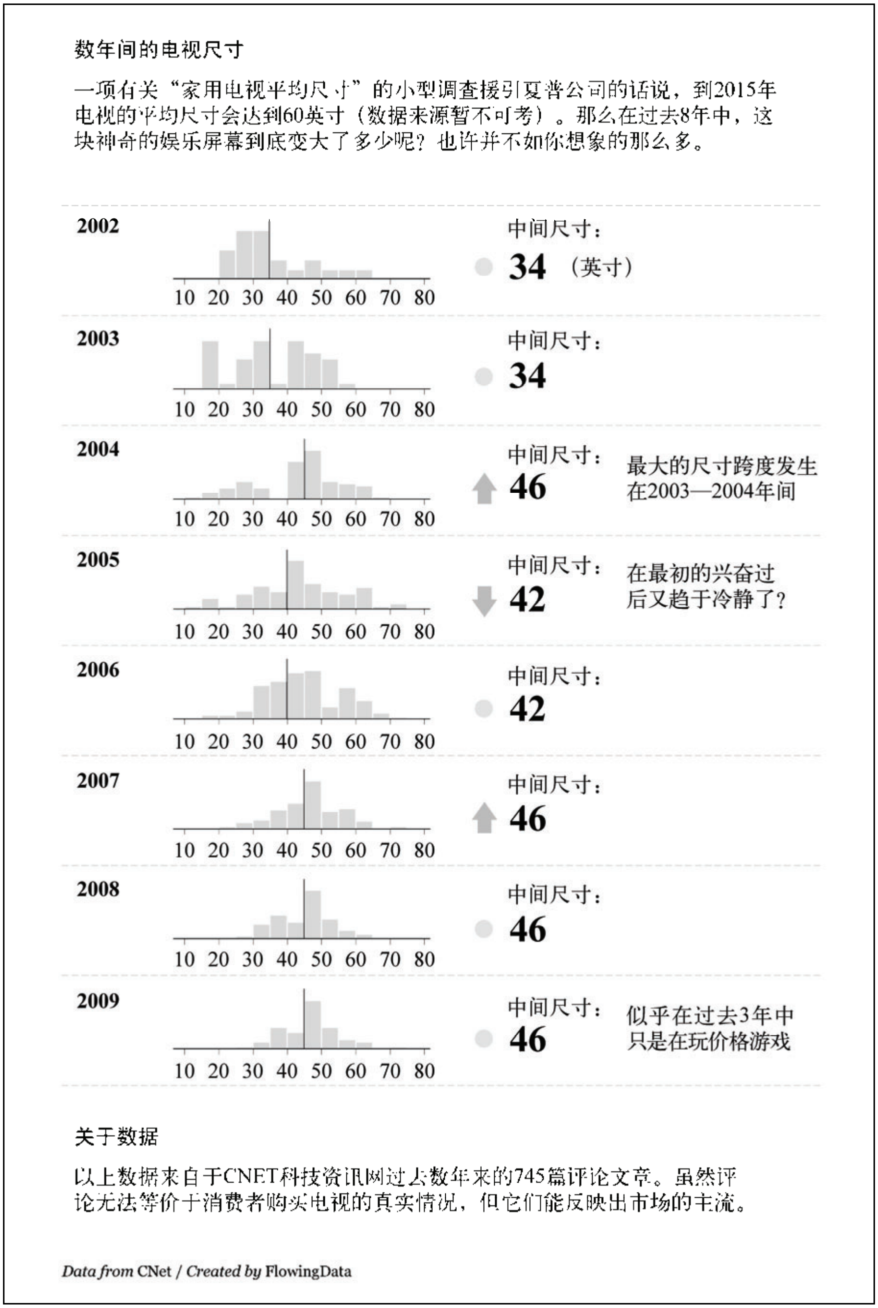


图6-36　数年间电视尺寸的分布

这段代码和我们之前写的有些出入，但总体逻辑是相似的：载入数据，过滤对异常值，然后画出一堆直方图。区别在于我没有用到lattice工具包中的`histogram()`函数。图表的布局用到了R中常见于全局参数设置的`par()`函数，而绘制直方图则用到了`hist()`函数。

```
# Load data
tvs <- read.table('http://datasets.flowingdata.com/tv_sizes.txt',
    sep="\t", header=TRUE)

# Filter outliers
tvs <- tvs[tvs$size < 80, ]
tvs <- tvs[tvs$size > 10, ]

# Set breaks for histograms
breaks = seq(10, 80, by=5)

# Set the layout
par(mfrow=c(4,2))

# Draw histograms, one by one
hist(tvs[tvs$year == 2009,]$size, breaks=breaks)
hist(tvs[tvs$year == 2008,]$size, breaks=breaks)
hist(tvs[tvs$year == 2007,]$size, breaks=breaks)
hist(tvs[tvs$year == 2006,]$size, breaks=breaks)
hist(tvs[tvs$year == 2005,]$size, breaks=breaks)
hist(tvs[tvs$year == 2004,]$size, breaks=breaks)
hist(tvs[tvs$year == 2003,]$size, breaks=breaks)
hist(tvs[tvs$year == 2002,]$size, breaks=breaks)
```

这段代码的输出如图6-37所示。图中有4行2列，这是由`par()`里面的`mfrow`参数来指定的。而在最终发布的图表里，我把它们放在一列显示。重要的是，用R可以让我避免在Illustrator或者Excel里面输入一大堆数据来手动生成8个图表。

2. 系列组图

将大量小图表归于一起的技巧通常被称作“系列组图”（small multiples）。这种图表方便读者多个群组和分类之间及其内部比较。此外，将有规律的多个图表整合到一起，还会显得更有组织性。

比如说，我曾经在烂番茄（Rotten Tomatoes）网站上浏览过很多“三部曲”系列影片的评分。可能有些读者不太熟悉烂番茄网，这个网站会聚合各种影片评论，并且将其划分为“好评”和“差评”。如果有60%以上的评论者声称他们喜欢某部影片，那么该影片就会被标记为“新鲜”。反之，自然就是“烂”了。我希望了解各大系列影片的续集和其首部曲相比，到底是更“新鲜”还是更“烂”。结论并不乐观，如图6-38所示。大结局得到的均分要比首部曲得到的均分低37%。换句话说，绝大多数首部曲都很“新鲜”，而绝大多数大结局都很“烂”。

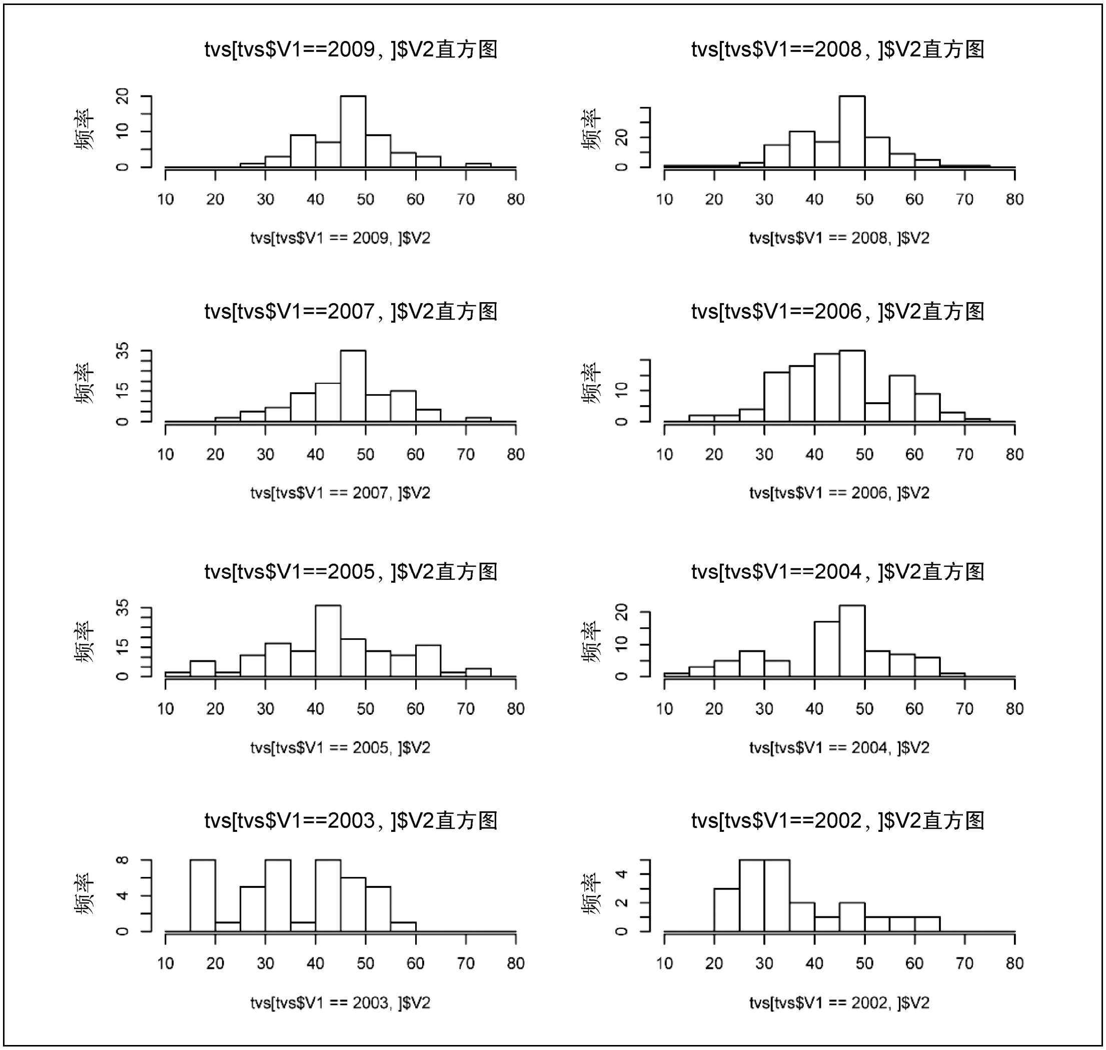


图6-37　直方图的网格布局

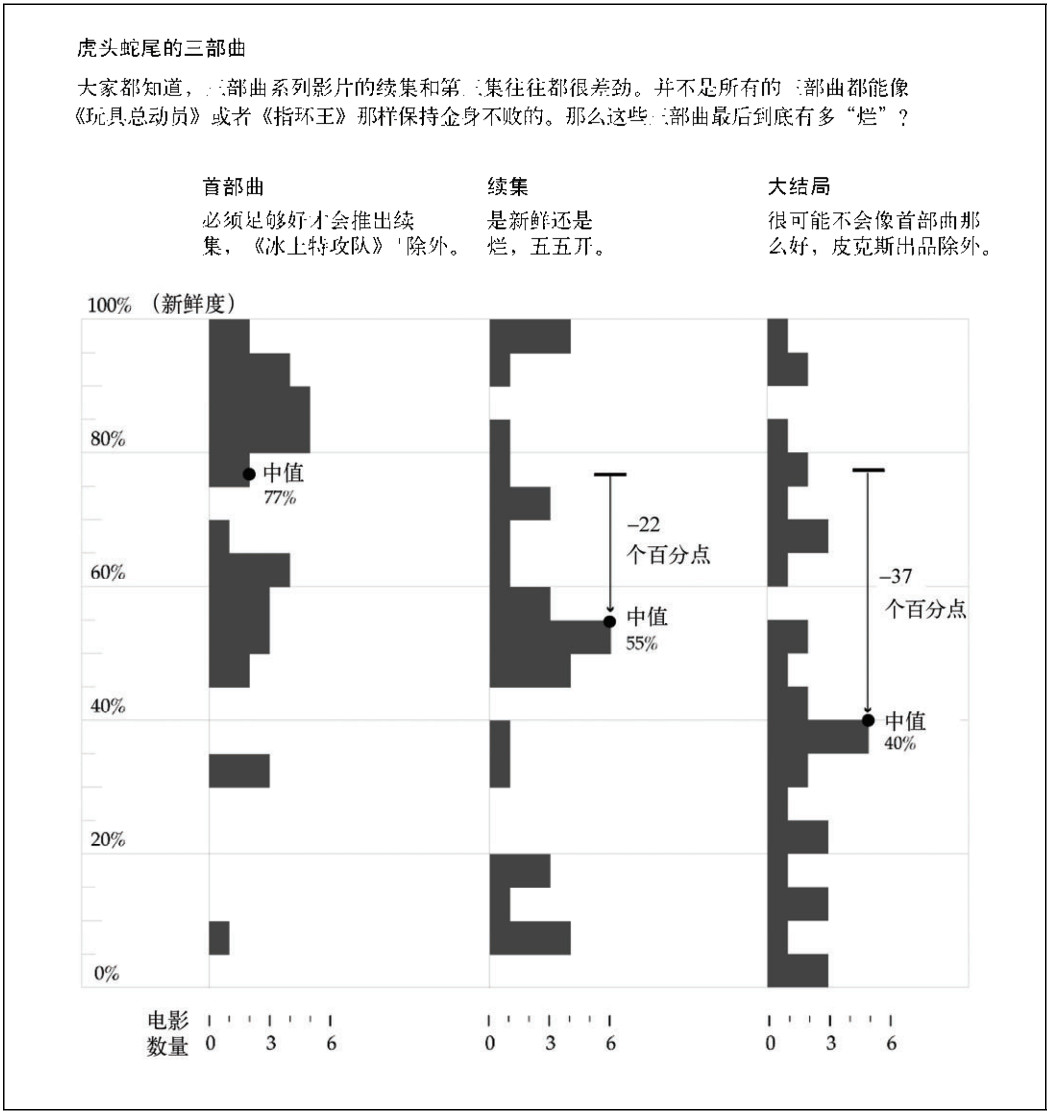


图6-38 三部曲系列影片从首部曲到大结局的评价趋势

实际上，这就是对3个直方图进行翻转得到的结果。图6-39显示了R中原始生成的直方图。我只是在Illustrator里面对它们进行了一些美化。

① 《冰河特工队》(*The Mighty Ducks* ）是迪士尼于1992、1994、1996年推出的三部曲系列影片，一气呵成、前后呼应，尤其是那些小童星从小演到大，是迪士尼非常成功的运动题材系列影片。

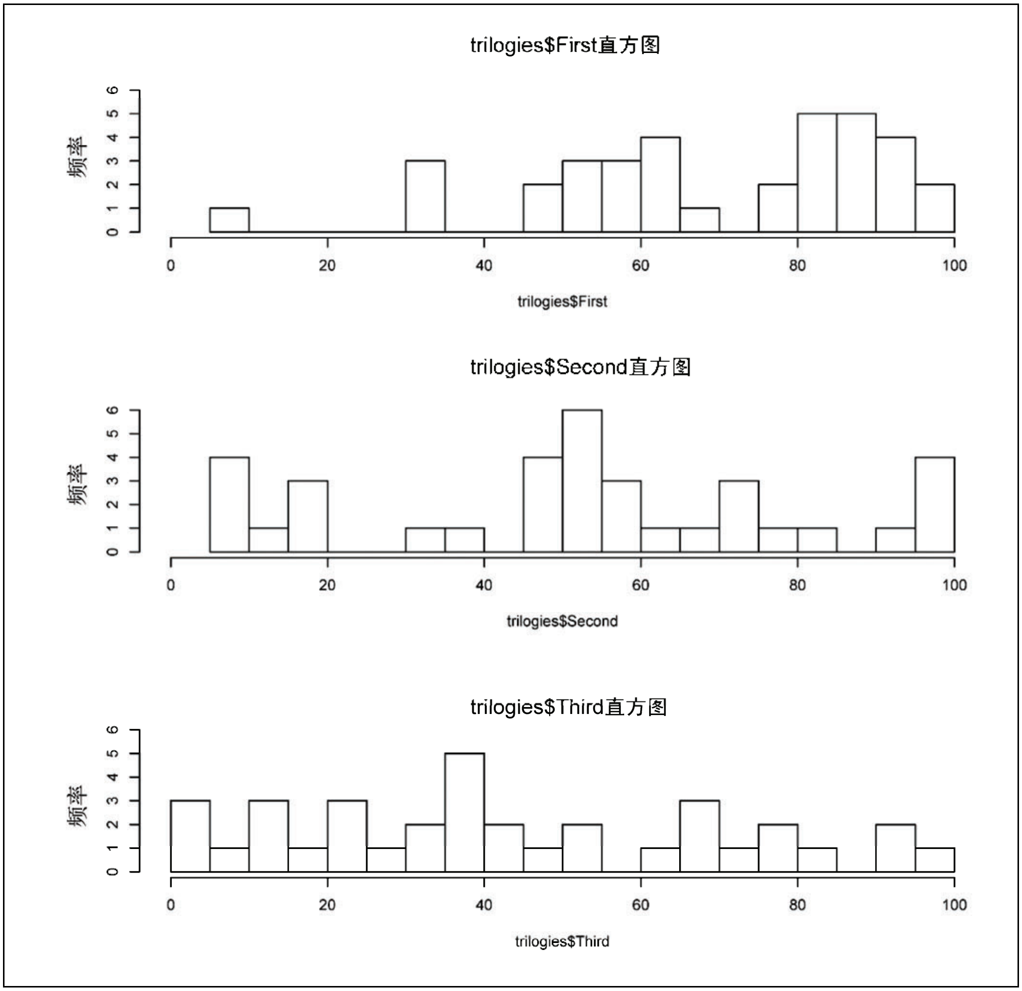


图6-39　原始的三部曲直方图

FlowingData的读者自然能够理解这些图表的意思——他们天生都对数据敏感。然而，这幅图表后来被IMDB，也就是网络电影资料库（Internet Movie Database）引用了。IMDB的受众群体要更为广泛，而且根据他们的评论来看，对数据不够敏感的读者在解读这些分布时非常吃力。

不过，图表的第二部分（见图6-40）看上去就容易理解多了。它运用了系列组图的概念，每一个柱形都代表一部电影的评分。红色的柱形代表“烂”，而绿色代表“新鲜”。

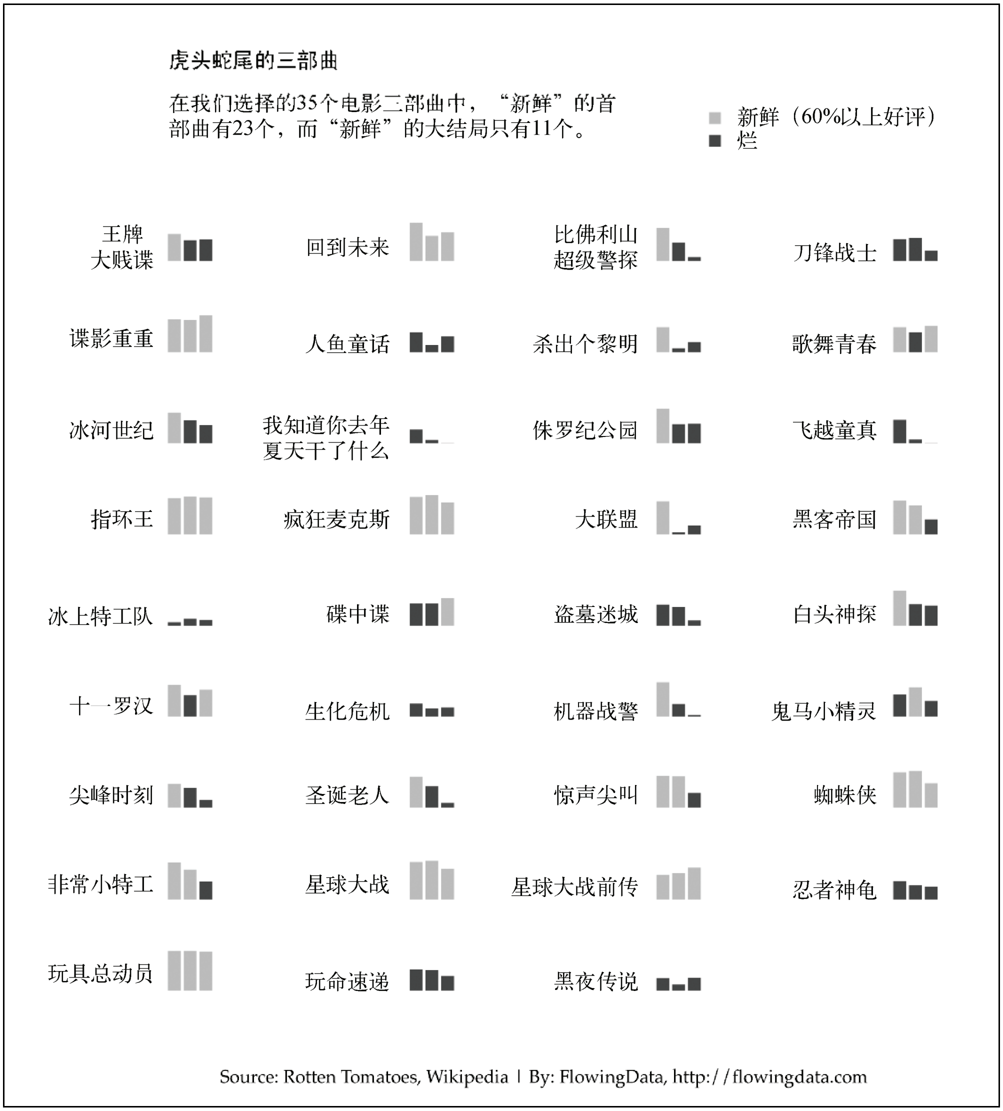


图6-40 三部曲评分的系列组图

可能有些人在想这是怎样做到的。它只是一大堆柱形图，我们可以像之前那样修改`mfrow`参数，并且用到`plot()`或者`polygon()`函数。不过我在这里用的是Illustrator里面的柱形图工具（Column Graph tool），因为当时它正好打开着。

在发表这幅图后，我学到了几件事情。首先也是最重要的是，集合与分布并不是每个人都能轻易理解的概念，因此我们需要尽力解释清楚，在讲故事时要倍加小心。我学到的第二件事情，就是人们忠于自己喜爱的电影，如果你对他们的挚爱横加批评，他们会很较真。

6.5　小结

在数据中探索关系有时候会颇具挑战性，而且还需要更多的批判性思维，不能盲目地拿着数字就开始画图。不过，这样也会呈现更多的信息，带来更大的回报。它表现了你的数据，或者说数据代表的事物之间是如何关联及互动的，这些才是最有意思的部分，正是它们造就了最好的故事。

本章讲述了如何在多个变量中找寻关联性，但同时也从更加宽泛的意义上阐述了关系的实质。从分布观察事物作为整体是如何彼此联系的，在分布中寻找模式、剔除异常值，然后考虑整件事的上下文背景。如果在这个过程中你发现了一些有趣的地方，尝试通过数据的上下文背景找到可能合理的解释。

这是在处理数据时最激动人心的部分，因为你开始探索数据的意义，而且很可能会挖掘到一些有意思的内容。当你挖掘得足够多了，就需要向读者解释自己的发现。记住，并不是每个人都能理解数字语言，因此请保持正常人能接受的水准。不过，如果你的读者都是数据迷，那么提高一下水准应该也没问题。

发现差异

7

体育评论员喜欢把少数运动员捧为超级明星，或者认为其高人一等，而把其他运动员视为平均水平或角色球员。这种分类一般并不是来自于统计数据，而是来自他们看过的大量比赛。这是种“我看即我识”的心理，并非满嘴跑火车。评论员（往往）都知道自己在说些什么，通常也会考虑到数据的上下文背景。每当体育分析员查看运动员的表现数据时，必定会有人提到：“你不能只看表面的数据。正是这种不确定性让赛事变得如此美妙。”这就是统计学的魅力所在。

很明显不只是体育界才这样。也许你希望找到街区最好的餐馆，或者找出谁是店里最忠实的顾客。在这种情况下，你也许不会去为个体进行分类，而是寻找那些从同类中脱颖而出的人或事。本章将讲述如何从全体中找出满足多种标准的集合，以及利用常识找出异常值。

7.1　在差异中寻求什么

如果只有一个变量，比较起来会很容易。这栋房子的占地面积比另外一栋要大，或者这只猫比另一只猫要重。如果有两个变量，可能会稍微困难一点，但依旧是可行的。第一栋房子的占地面积比较大，但第二栋房子的浴室数量更多。第一只猫更重、毛更短，而第二只猫较轻、毛较长。

但如果有一百栋房子或者一百只猫，该怎样分类呢？如果每栋房子都有更多变量，例如卧室的数量、后院的面积还有物业管理费，又该怎么办？这时我们面对的是对象的总数乘以变量的总数。现在的情况要棘手多了，而这正是我们所关注的。

也许你的数据中包含多种变量，但你希望把所有对象（比如人或地点）进行分组，然后找出其中最出众的或者说异常值。你希望看到每一个变量间的差异，也希望看到所有变量间的差异。两个篮球运动员的场均得分可能是天壤之别，但他们的场均篮板、抢断和出场时间却可能非常接近。我们需要找出事物间的差异，但也不要忘记相似性和背后的关联性，就和那些体育评论员一样。

7.2　在多个变量间比较

在面对多个变量时，主要的挑战在于从哪里开始。如果我们不事先考虑好自己掌握的数据是什么，在面对如此众多的变化及子集时就会无所适从。有时候，最好的做法是先一次性观察所有的数据，那些让你感兴趣的内容自然会为你指引方向。

7.2.1 热身

要想对表格中的数据进行可视化，最直接的方法就是一次性把它们都显示出来。不过我们可以用颜色代替数字来表示数值，如图7-1所示。

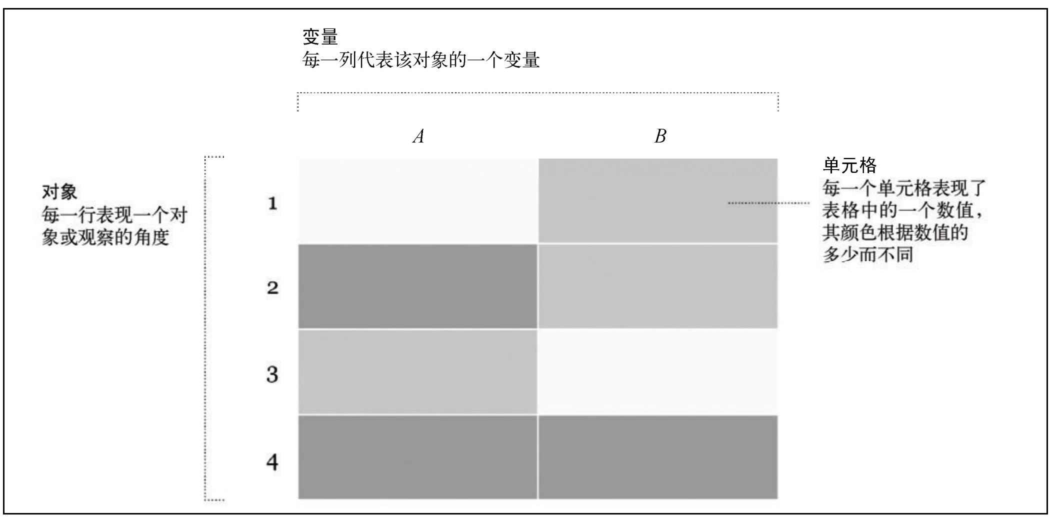


图7-1　热点图的基本框架

我们会得到一个与原始数据表格大小相同的网格，但根据单元格颜色的不同，很容易就能找出那些相对较高或较低的数值。一般来说，深色代表高数值，浅色代表低数值。不过根据应用场景不同，这一规则也有可能会发生变化。

在阅读热点图（heatmap，或者叫热点矩阵，heat matrix）时，使用的方法也与阅读表格时相同。我们可以从左到右地阅读，了解一个对象的所有变量的数值，也可以只选择一个变量，了解所有对象在该方面的表现如何。

这种布局可能依旧会让人困惑，尤其是当数据表格非常庞大时。但有了颜色的区分以及排序，得到的图表还是能为我们带来很大帮助。

创建热点图

在R里面创建热点图非常简单。有`heatmap()`函数来包办所有的计算工作，我们只需选择合适的颜色，然后组织好标签，以便在有多行多列的情况下保持易读性。换句话说，R负责基础框架，而我们负责设计。这本书读到这里，这句话大家应该很耳熟了吧。

在本例中，我们来看一下2008赛季NBA球员的统计数据。大家可以在http://datasets.flowingdata.com/ppg2008.csv下载到CSV格式。数据共有22列，第一列是球员姓名（Name），其他列则是统计数据，例如PTS（场均得分）和FGP（投篮命中率）。你可以在R里用`read.csv()`来载入数据。现在让我们看看数据的前五行，以便对数据的结构有个概念（见图7-2）。

```
bball <-
    read.csv("http://datasets.flowingdata.com/ppg2008.csv",
    header=TRUE)
bball[1:5,]
```

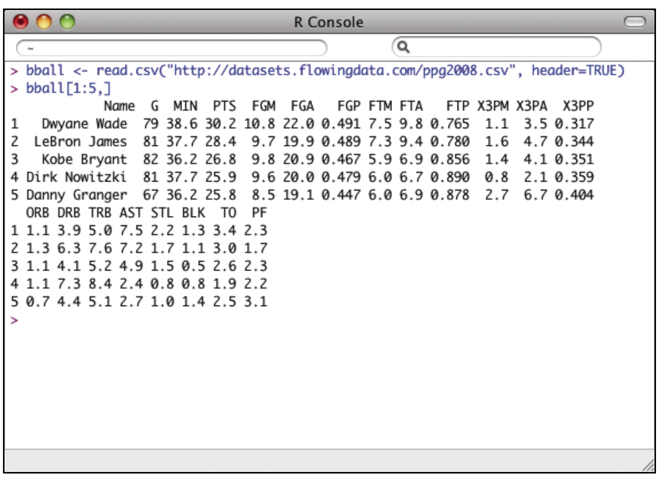


图7-2 前五行数据的结构

现在这些球员是按PTS由高到低排序的。但我们也可以通过`order()`函数来按任一列排序，例如场均篮板数（TRB）或者FGP。

```
bball_byfgp <- bball[order(bball$FGP, decreasing=TRUE),]
```

提示　order()中的decreasing参数可以指定是按升序还是降序排列。

现在再看看bball_byfgp的前五行，我们会发现刚才排名前列的Dwyane Wade（德怀恩·韦德）、Lebron James（勒布朗·詹姆斯）和Kobe Bryant（科比·布莱恩特）被换成了Shaquille O'neal（沙奎尔·奥尼尔）、Dwight Howard（德怀特·霍华德）和Pau Gasol（保罗·加索尔）。在本例中，我们将根据PTS逆序排列。

```
bball <- bball[order(bball$PTS, decreasing=FALSE),]
```

现在，每一列的名称与CSV文件的数据头是一致的，这和我们想要的相符。不过，我们还希望每一行的名称应该是球员的姓名，而不是现在的行号。所以需要把第1列设置为行名称。

```
row.names(bball) <- bball$Name
bball <- bball[,2:20]
```

第一行代码将行名称改成了数据块中的第1列。第二行代码选择了第2列到第20列，并将该子集设置为新的bball。

另外，数据必须是矩阵（matrix）格式，而不能是数据块（data frame）格式。如果我们在heatmap()函数中使用数据块就会报错。总的来说，数据块有点像一系列向量的集合，其中每一列都代表一个不同的度量。每一列都可以有不同的格式，例如数字或者字符串。而矩阵则通常被用于表现二维空间，而且所有单元格中的数据类型必须保持统一。

```
bball_matrix <- data.matrix(bball)
```

提示　很多可视化都涉及数据的收集和准备。我们获得的数据往往和需要的并不完全相符，因此大家要有心理准备，在进入视觉部分之前可能得做一些数据再加工。

现在数据已经按我们所希望的进行了排序及格式化，将其放入heatmap()里面就能有所收获了。我们把scale参数指定为"column"，这是在告诉R用每一列的最大值和最小值来定义颜色的色阶，而非用整个矩阵的最大值和最小值。

```
bball_heatmap <- heatmap(bball_matrix, Rowv=NA,
    Colv=NA, col = cm.colors(256), scale="column", margins=c(5,10))
```

你的结果应该类似于图7-3。我们通过cm.colors()指定了颜色范围由青色（cyan）变为洋红色（magenta）。这一函数创建了一个16进制的颜色向量，范围默认由青到洋红，其中有*n*个不同的色值（本例中是256个）。所以，请注意图中的第3列（也就是PTS）由洋红色开始，这表示Dwyane Wade和Lebron James在该方面的数值是最高的，然后逐渐变色，到底部已经变成了较暗的青色系，代表该方面数值置底的Allen Iverson和Nate Robinson。我们也可以很快找到其他洋红色的单元格，例如篮板数（TRB）最高的Dwight Howard或者助攻（AST）最突出的Chris Paul。

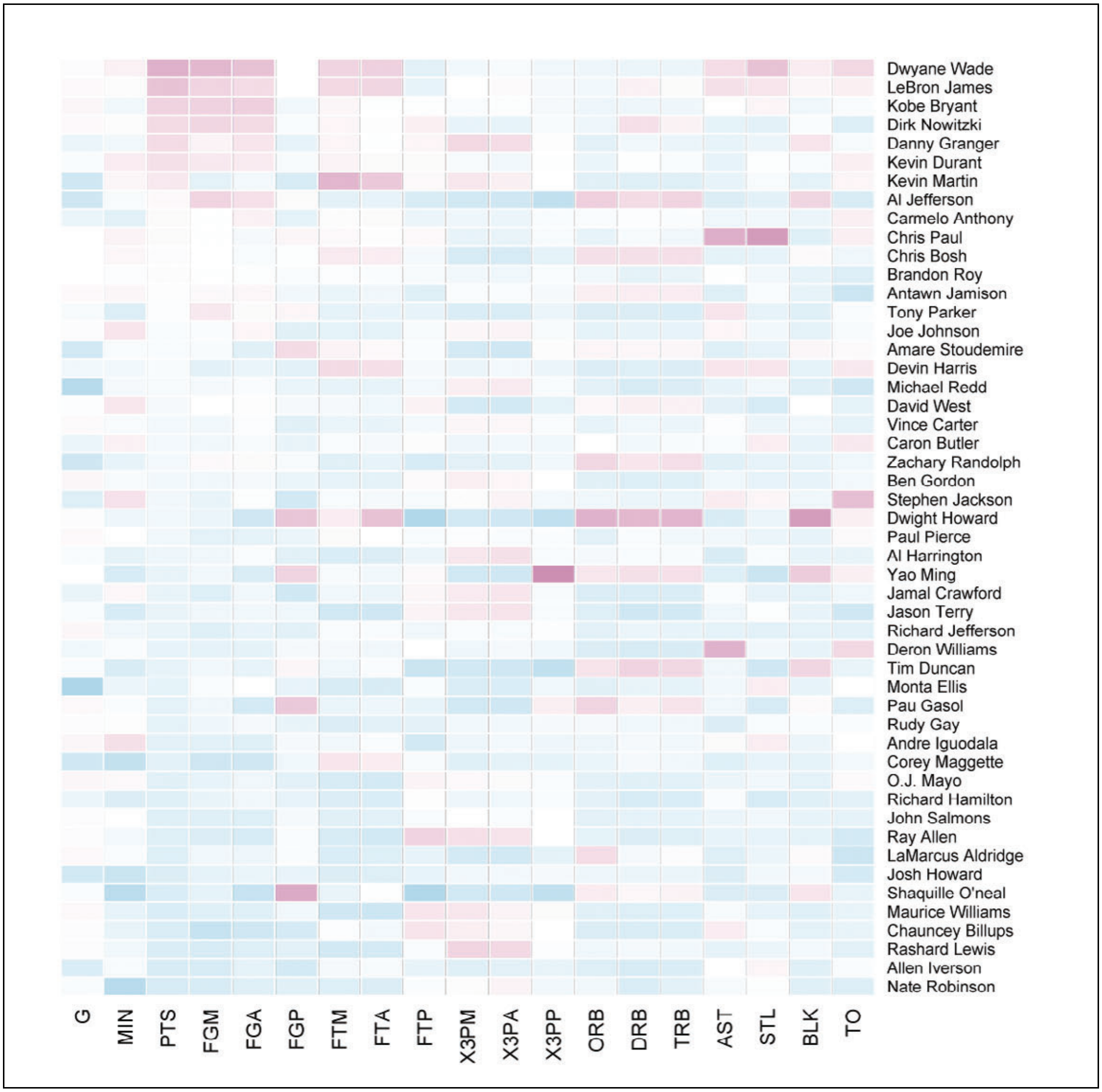


图7-3 按PTS排序的默认热点图（另见彩插图7-3）

你也许想要换一种颜色风格。只需要改动col参数即可，在我们刚才运行的那段代码中，它是cm.colors(256)。可以输入?cm.colors寻求帮助，看看R都提供哪些颜色。比如我们可以用看上去更有激情的颜色，如图7-4所示。

```
bball_heatmap <- heatmap(bball_matrix,
    Rowv=NA, Colv=NA, col = heat.colors(256), scale="column",
    margins=c(5,10))
```

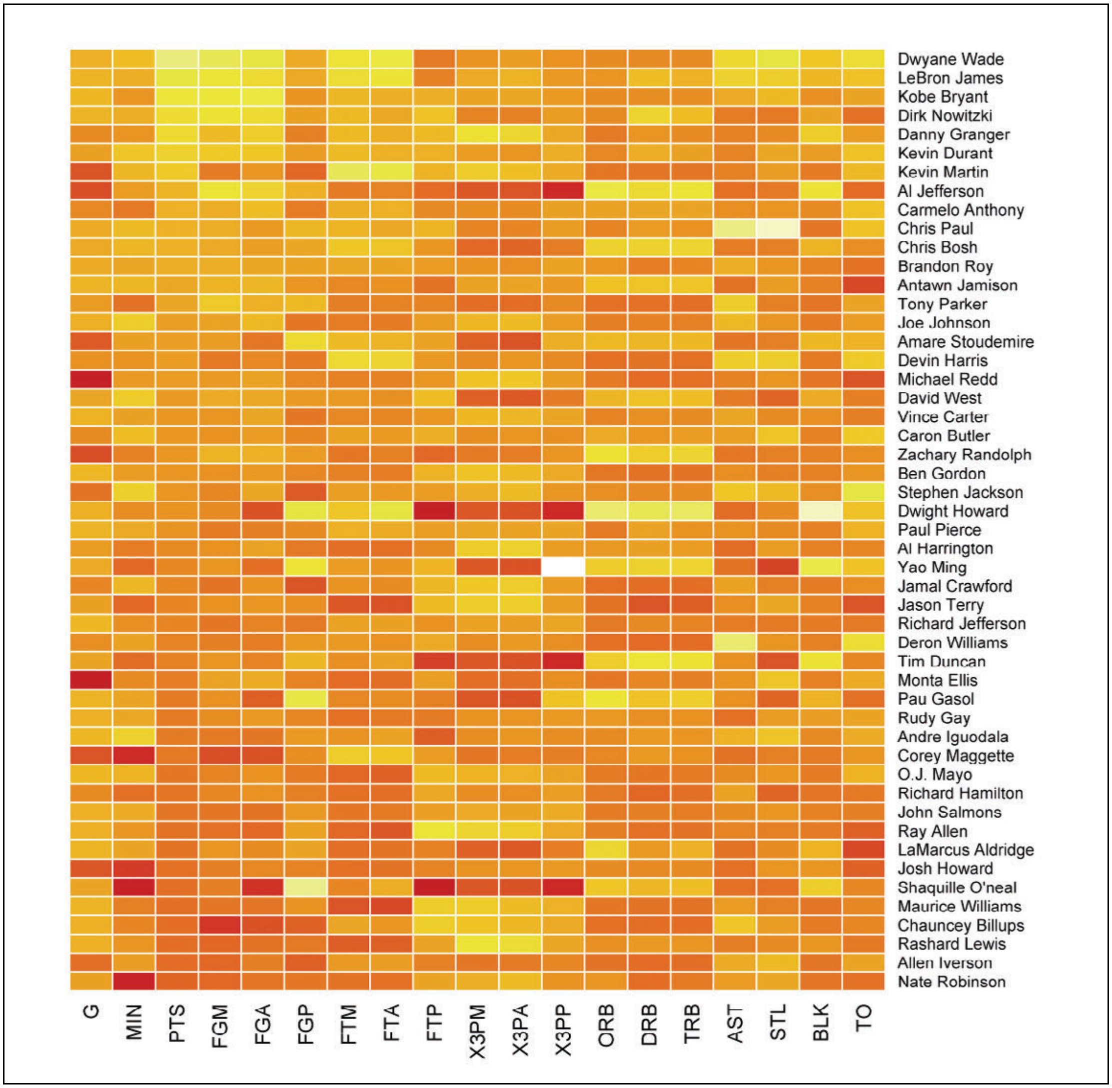


图7-4　红–黄色系的热点图（另见彩插图7-4）

如果你在R中输入`cm.colors(10)`，就会得到一个由青到洋红的10个颜色的数组。之后`heatmap()`会自动按线性运算，为各个数值选择相应的颜色。

```
[1] "#80FFFFFF" "#99FFFFFF" "#B3FFFFFF" "#CCFFFFFF" "#E6FFFFFF"
[6] "#FFE6FFFF" "#FFCCFFFF" "#FFB3FFFF" "#FF99FFFF" "#FF80FFFF"
```

这一点很棒，因为我们可以轻松地创建自己的颜色范围。比如说，我们可以去0to255.com先选择一种基本色，然后再详细选择。图7-5显示了基本色为红色的色阶。我们可以选出一些颜色，由浅到深，然后放到`heatmap()`里面，如图7-6所示。这次我没有让R来创建颜色向量，而是定义

了一个自己的red_colors变量。

```
red_colors <- c("#ffd3cd", "#ffc4bc", "#ffb5ab",
    "#ffa69a", "#ff9789", "#ff8978", "#ff7a67", "#ff6b56",
    "#ff5c45", "#ff4d34")
bball_heatmap <- heatmap(bball_matrix, Rowv=NA,
    Colv=NA, col = red_colors, scale="column", margins=c(5,10))
```

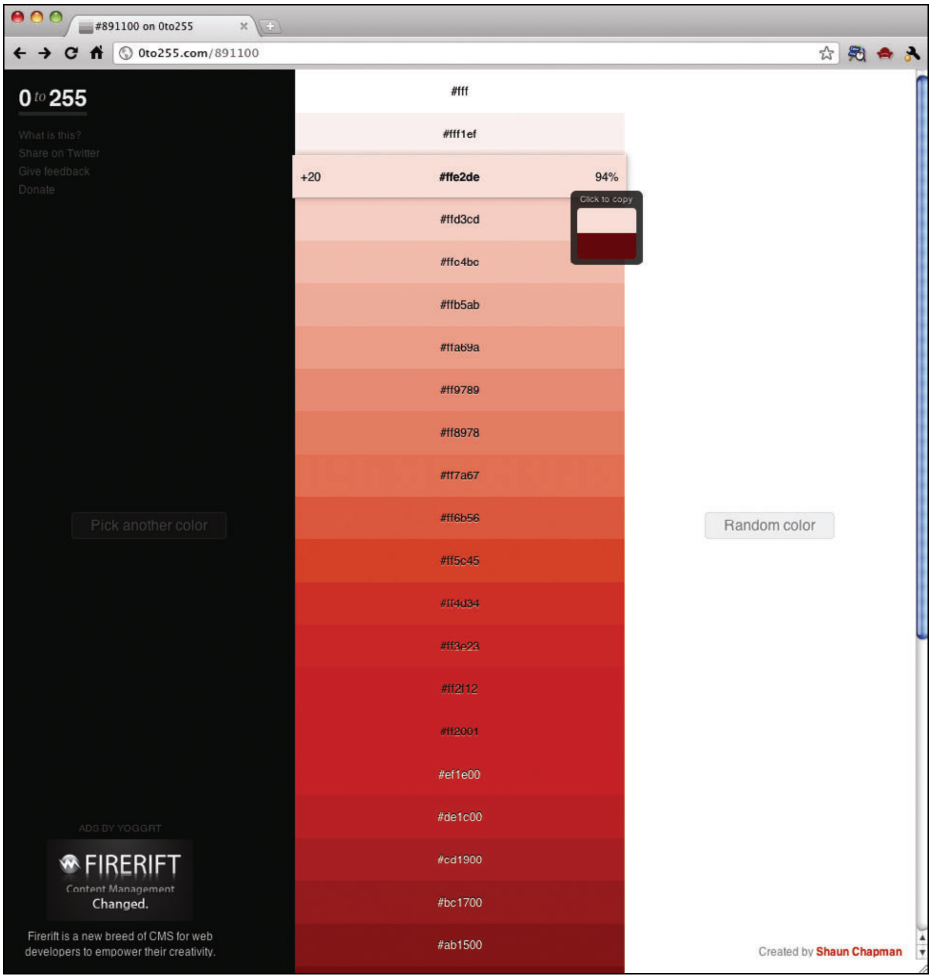


图7-5　0to255.com里面的红色系色阶（另见彩插图7-5）

提示　明智地选择颜色，因为它们也会奠定你的故事的整体基调。比如说，如果你的主题较为严肃，可能中性、柔和的色调会更好，而激昂、休闲的主题则更适合采用明亮的色彩。

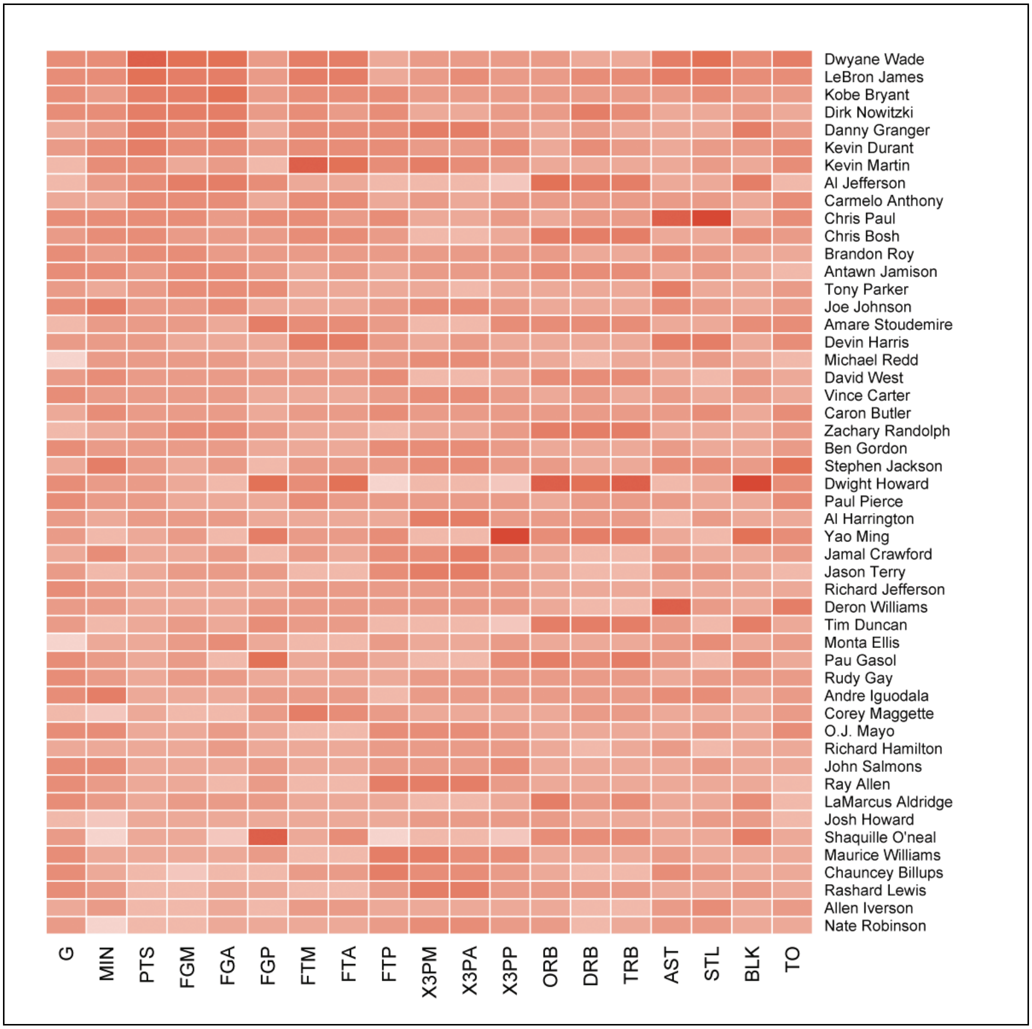


图7-6　使用了自定义红色系的热点图（另见彩插图7-6）

如果你不想自己设定颜色，也可以使用R的ColorBrewer工具包。这个工具包不是默认安装的，因此如果你没有的话，就需要下载并通过Package Installer来安装。ColorBrewer是由制图师Cynthia Brewer设计的，本来用于地图绘制，不过它也能帮助我们创建常用的数据图表。有很多选项可供选择，例如各种连续色或相反色的调色盘以及许多色阶等。本例中我们选择了简单的蓝色调色盘。在R中输入`?brewer.pal`可以得到更多选项，非常好玩。假设你安装了RColorBrewer，输入以下代码就能得到一个蓝色系、9色阶的热点图，效果如图7-7所示。

```
library(RColorBrewer)
bball_heatmap <- heatmap(bball_matrix, Rowv=NA,
    Colv=NA, col = brewer.pal(9, "Blues"),
    scale="column", margins=c(5,10))
```

> ▶ 访问http://colorbrewer2.com以尝试ColorBrewer的可交互版本。你可以在下拉菜单里进行选项设置，在样本地图中看到各种颜色主题的效果。

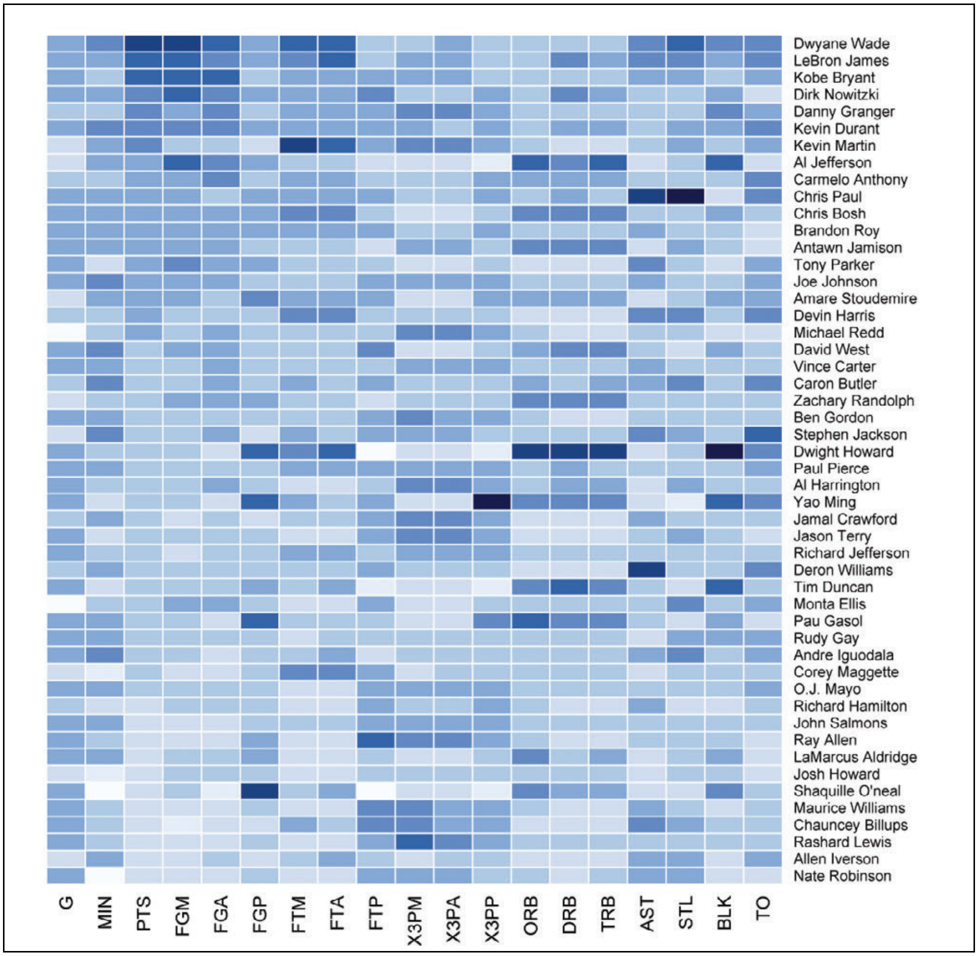


图7-7　利用RColorBrewer作为调色盘的热点图（另见彩插图7-7）

图7-7还可以再导入到Illustrator里面继续优化。图形方面不需要太多修饰，但我们可以让标签更加易读一些、颜色更加柔和一些，让图表更加易于浏览。

对于标签来说，最好不要用缩写。作为篮球迷来说，我知道每一个缩写都代表什么意思，但那些对这项运动不太熟悉的人可能会摸不着头脑。对于颜色，我们可以打开Illustrator的“颜色”（Color）面板，通过调整透明度来缓和对比度。为单元格添加边框也能提升可辨识度，让图表更加易于从左到右、从上到下地浏览。图7-8显示了最终完成的效果。

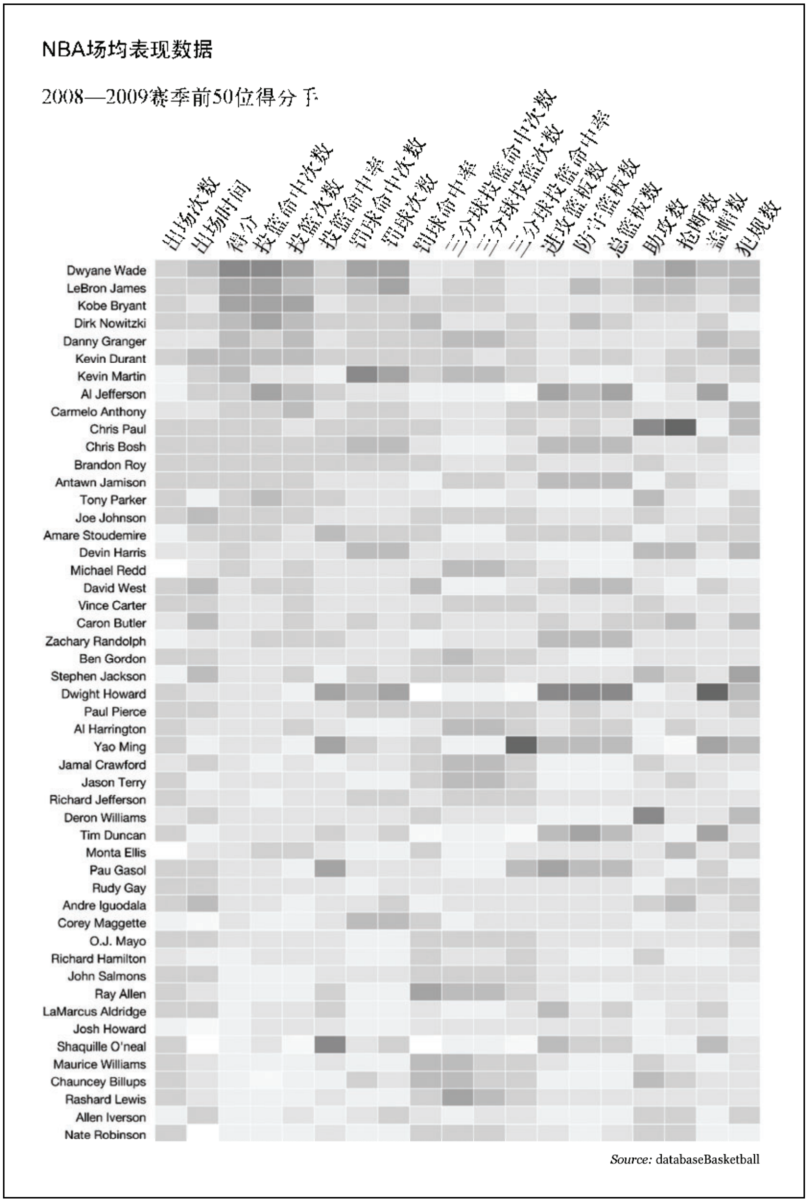


图7-8　显示2008—2009赛季NBA前50位得分手场均表现数据的热点图

▶ Mike Bostock在Protovis里也借用了我的这个例子，大家可以在Protovis的范例部分找到。视觉上两者完全一样，但他的范例的交互效果更加丰富：当鼠标移到单元格上时会出现提示信息。

7.2.2　相面术

热点图的好处在于它能让我们一次性看到所有的数据，但是这样一来，关注点总会落在单个的点上。我们可以很快找出场均得分或者篮板数的高低，但要想将某位球员和另一位球员进行综合比较，却存在着很大难度。

有些时候，我们不希望让每个对象都被各种指标切散，而是希望把它们当做一个个的整体来观察。切尔诺夫脸谱图（Chernoff Faces）就能满足这种需求。该方法并不是业界通用的标准方法，而且很可能会让那些普通读者一头雾水。但尽管如此，切尔诺夫脸谱图还是经常能体现出其价值的。另外，它对于数据迷来说非常有趣，我想这个理由就足够了。

切尔诺夫脸谱图的关键在于，它会根据数据集中的数字将多个变量一次性展现在人脸的各个部位上，例如耳朵、头发、眼睛、鼻子等（见图7-9）。我在此假设读者在现实生活中都能轻易辨识人的面部特征，这样当它们在表现数据时，大家才可能识别出那些微小的差别。这是个比较冒险的假设，但我相信大家都有这种能力。

大家在后面的例子中将会看到，较大的数值会以更多的头发或更大的眼睛的形式来出现，而较小的数值则会对面部特征进行收缩。除了尺寸大小以外，我们还可以调整诸如嘴唇曲线或脸型等其他面部特征。

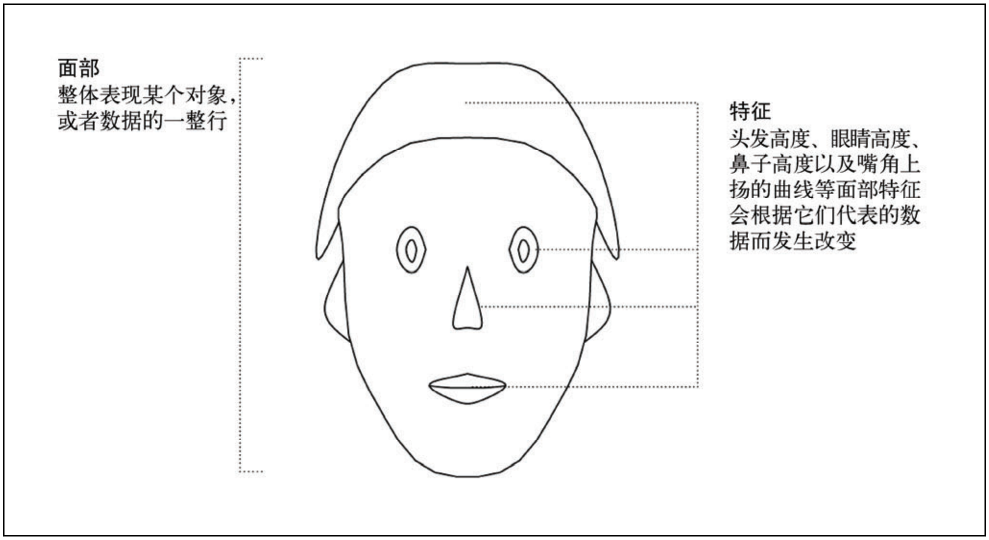


图7-9　切尔诺夫脸谱图的基本框架

创建切尔诺夫脸谱图

让我们回到之前的篮球数据，也就是2008—2009赛季NBA前50位得分手的统计成绩。每一位球员都会有一张脸谱。不用慌张——你不必手动去画这些脸。R中的aplpack提供了faces()函数，它能帮助我们得到想要的结果。

如果你还没有这个工具包，可以通过install.packages()来安装，或者利用Package Installer选项来完成。这个工具包的名称aplpack代表的意思是another plotting package（另一个绘图工具包），由Hans Peter Wolf设计。安装完成后，工具包应该会自动载入，但有可能也需要手动载入。

```
library(aplpack)
```

说明　最新版本的aplpack能让我们通过faces()函数来添加颜色。在本例中，我们将ncolors设置为0，也就是只使用了黑色和白色。输入?faces可以查阅如何使用颜色向量，其方法和之前热点图例子中的用法是一样的。

在之前创建热点图时，你应该已经载入了篮球数据。如果没有的话，再次使用read.csv()通过URL来直接载入数据。

```
bball <- read.csv("http://datasets.flowingdata.com/ppg2008.csv",
header=TRUE)
```

载入了工具包和数据之后，就可以直接用faces()函数来生成切尔诺夫脸谱图了，结果如图7-10所示。

```
faces(bball[,2:16], ncolors=0)
```

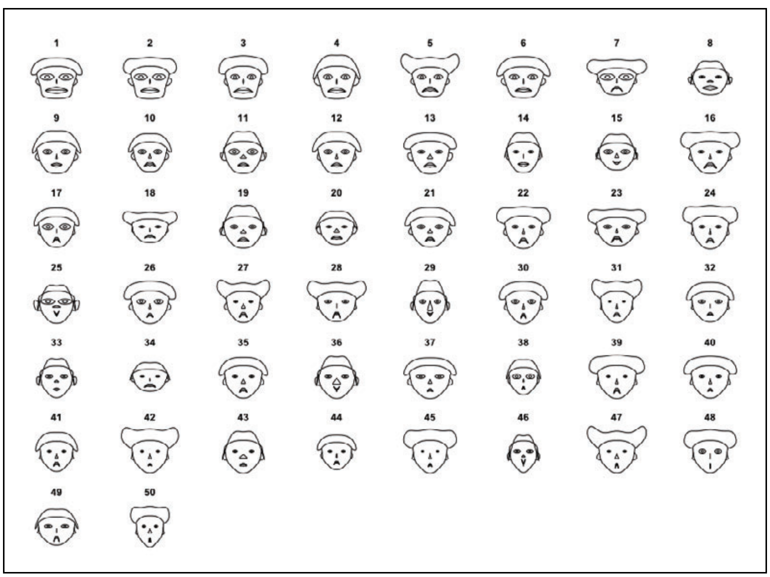


图7-10　默认生成的切尔诺夫脸谱图

算上球员的姓名，我们的数据集中总共有20个变量。但是aplpack提供的faces()函数只支持最多15个变量，因为只有这么多脸部特征能够被改变。这就是为什么我们要将集合缩小到从第2列到第16列。

每张脸谱都代表什么意思呢？faces()函数会配合数据集中各列的顺序，按照以下顺序来修改面部特征。

(1) 脸庞的长度

(2) 脸庞的宽度

(3) 脸型

(4) 嘴唇的高度

(5) 嘴唇的宽度

(6) 嘴角上扬的曲线

(7) 眼睛的高度

(8) 眼睛的宽度

(9) 头发的高度

(10) 头发的宽度

(11) 发型

(12) 鼻子的高度

(13) 鼻子的宽度

(14) 耳朵的宽度

(15) 耳朵的高度

所以，脸庞的长度代表的是赛季出场的次数，而嘴唇的高度则代表场均投篮命中的次数。由于每张脸都没有对应的姓名，这张图表似乎没多大用处，不过我们还是能看出头几名球员各方面都表现比较全面，而第7名球员的头发相对很宽，这表示他每场投中的三分球较多。

提示 如果对应的个体很多，对其进行分类可能会便于对脸谱的浏览。比如说，我们可以根据球场上的位置来为这些脸谱进行分类：后卫、前锋和中锋。

7

在faces()中使用labels参数来为脸谱添加姓名，结果如图7-11所示。

```
faces(bball[,2:16], labels=bball$Name)
```

这样看上去好多了。现在我们可以看到哪张脸对应哪位球员。要想找出控球后卫，我们可以从Chris Paul（克里斯·保罗）开始，找寻和他脸谱相似的球员，例如Devin Harris（德文·哈里斯）或者Deron Willams（德隆·威廉姆斯）。不过，右下角的Chauncey Billups（昌西·比卢普斯）也是一位控球后卫，但他的脸看上去与其他控球后卫不太一样。他的头发更高，嘴唇更窄，这些表示他的罚球命中率较高，而场均投篮次数较少。

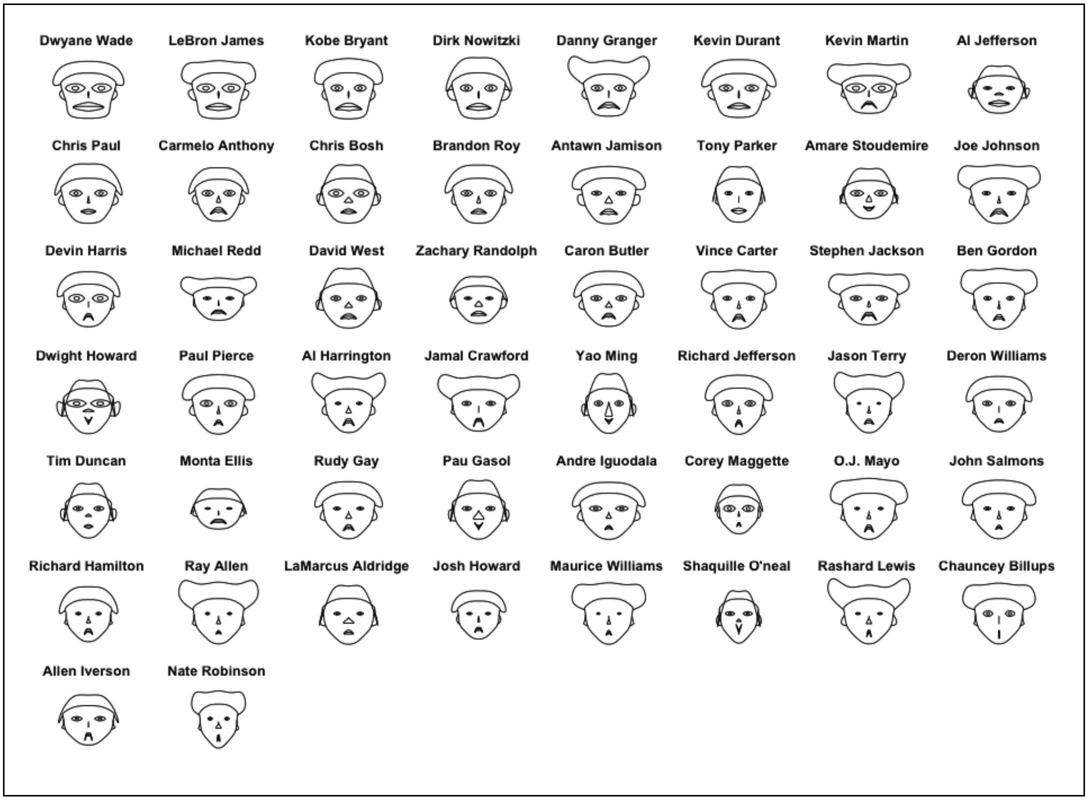


图7-11　带有球员姓名的切尔诺夫脸谱图

为了让图表更加易读，我们可以加大各行之间的距离，或者至少提供一段描述，说明哪些面部特征代表哪些数据，如图7-12所示。通常情况下，我会用一张实际图例来进行说明，但我们用到的面部特征实在太多，在一张脸上画那么多箭头实在是惨不忍睹。

提示　在画面中留出一些空白区域，会让图表更加易读，尤其是在要看和评估的内容很多的时候。

再次申明，切尔诺夫脸谱图的价值会根据数据集与受众群体的不同而不同，因此大家可以自行决定是否使用这一方法。还有一点是，那些不太熟悉切尔诺夫脸谱图的人总喜欢“以貌取人”，他们认为代表Shaquille O’Neal的脸谱似乎就应该与他本人类似。但我们绝大多数人都知道，O’Neal可是有史以来体型最大的篮球运动员之一。

用同样的方式，我又设计了美国犯罪情况的切尔诺夫脸谱图，如图7-13所示。居然有人因为那些高犯罪率州的脸型而评论说它有种族歧视倾向。我可从没想过这一点，因为对我来说，修改某个面部特征就和修改某个柱形的长度一样。不过这个问题倒的确值得考虑一下。



图7-12　2008—2009赛季NBA前50位得分手的切尔诺夫脸谱图

提示　在设计图表时，要站在普通读者的角度考虑问题。普通读者并不像我们那样熟悉各种可视化方式，对数据的理解也不如我们深刻，因此作为故事的讲述者，我们必须承担起解释说明的责任。



图7-13　美国犯罪率统计，每张脸谱代表一个州

7.2.3　星光灿烂

除了用脸谱来表现多变量数据之外，我们还可以遵循同样的概念将图表抽象为另一种形状。

我们不再修改面部特征，而是直接修改形状来表现不同的数据值。这就是星图（star chart）的制作原理，它也被称为雷达图（radar chart）或者蜘蛛图（spider chart）。

如图7-14所示，我们可以绘制多条轴，每一条轴代表一个变量，由正中心开始，等距平分圆周摆放。正中心表示各个变量的最小值，而轴末端的终点代表最大值。在绘制图表时，在每个变量的终点和下一条轴的终点之间还需要绘制一条连接线。最后得到的图表就好像一颗星星一样（或者说像雷达或蜘蛛网）。

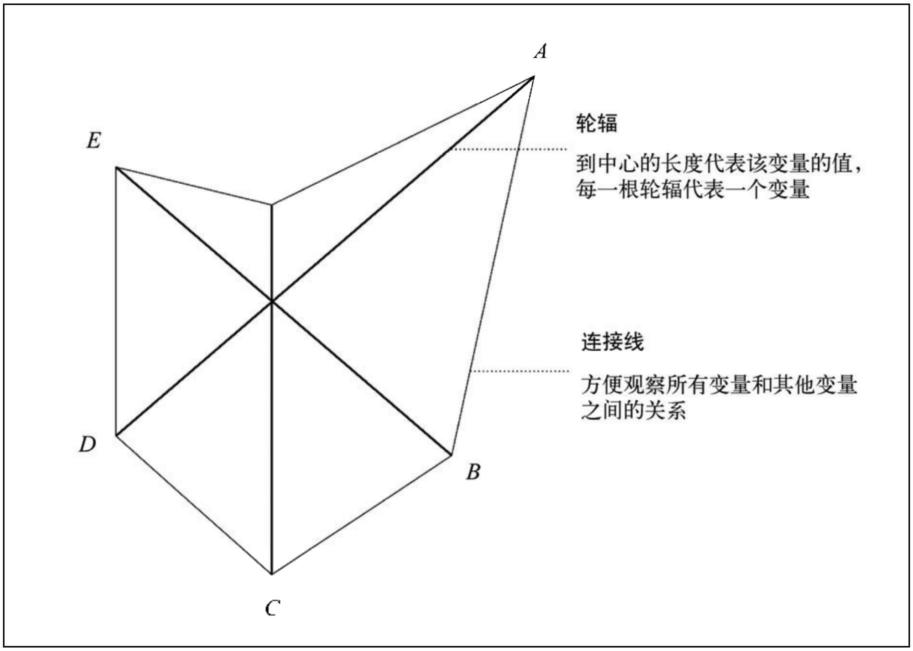


图7-14　星图的基本框架

我们也可以在一幅星图里表现多个对象，但如果数量稍多就会过于杂乱，导致故事没法讲清楚。所以最好还是坚持用一个星图表现一个对象，这样更便于比较。

创建星图

现在让我们继续使用图7-13用到的犯罪数据，来看看星图的表现手法会有哪些不同。首先还是在R中载入数据。

```
crime <- read.csv("http://datasets.flowingdata.com/
crimeRatesByState-formatted.csv")
```

然后就可以直接创建星图了，和创建切尔诺夫脸谱图非常类似。我们用到的是R中已封装好的`stars()`函数。

```
stars(crime)
```

7

默认的图表如图7-15所示。希望这幅图不会冒犯任何人。自然，每个州的标签尚待添加，此外还需要一个图解（key），以便了解每一条轴所代表的意义。和`faces()`一样，这些轴肯定遵循着某种顺序，但我们无法知道第一个变量是从哪里开始的。所以让我们一次搞定这些内容。别忘了可以把第一列移动到行名称上去，就和热点图中我们所做的一样。此外，还可以把`flip.labels`设置为`FALSE`，因为我们不希望标签影响到高度。结果如图7-16所示。

```
row.names(crime) <- crime$state
crime <- crime[,2:7]
stars(crime, flip.labels=FALSE, key.loc = c(15, 1.5))
```

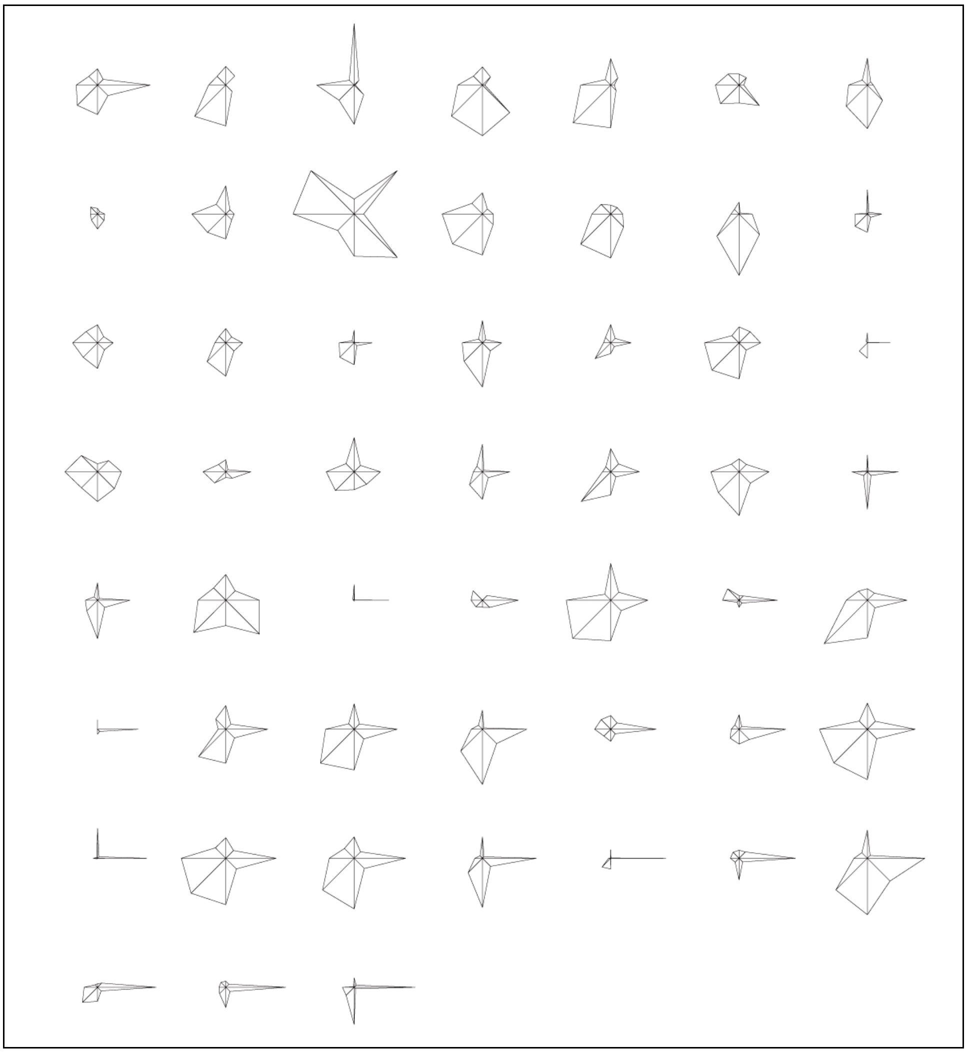


图7-15　显示各州犯罪情况的默认星图

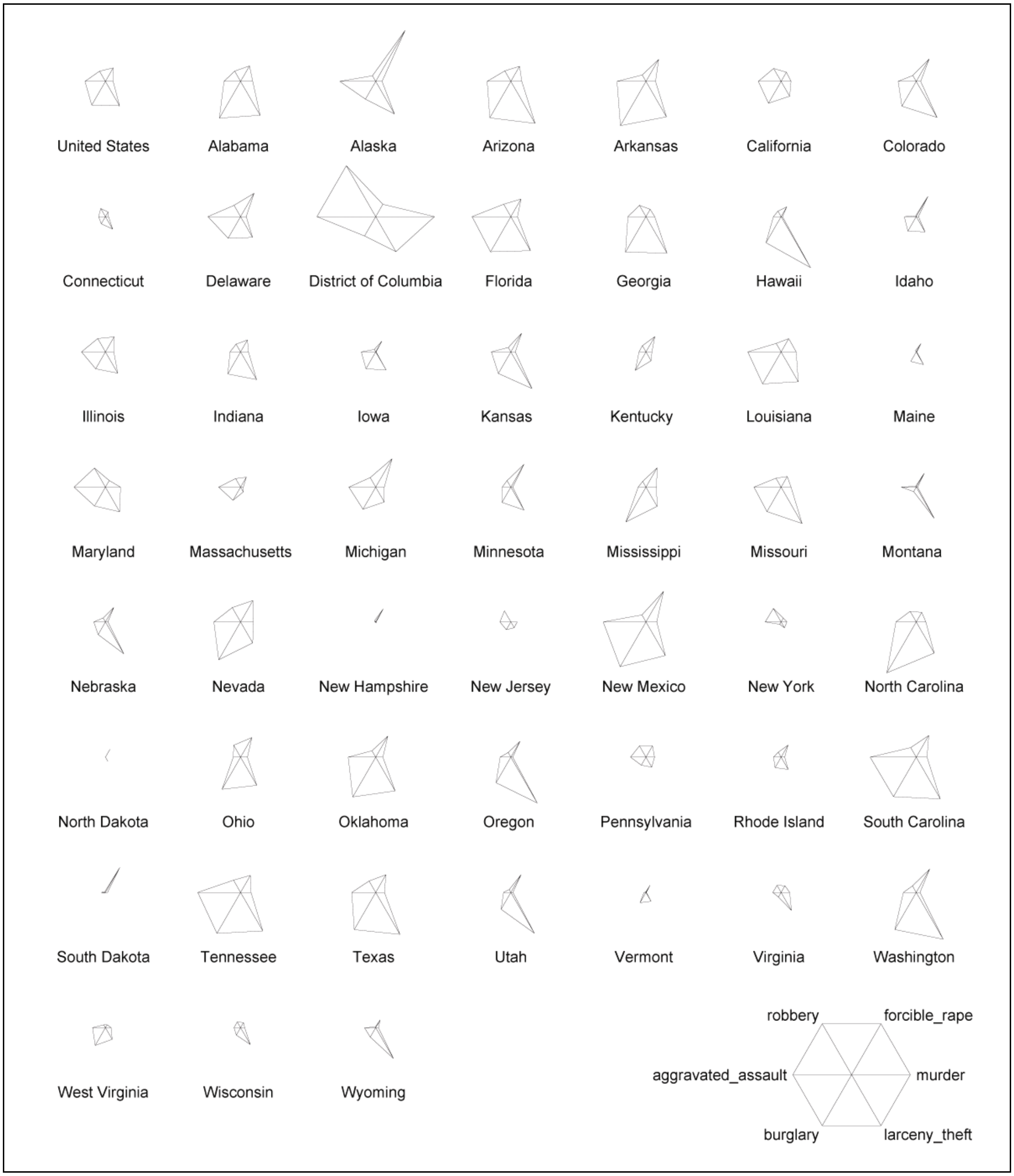


图7-16　添加了标签和说明的星图

现在更容易辨别各州的差别和相似之处了。在切尔诺夫脸谱图中，哥伦比亚特区（District of Columbia）看上去就像一个大眼睛小丑，而在星图中我们看到，尽管它某些类型的罪案发生率较高，但强奸和入室盗窃的发生率还是相对较低的。我们也很容易找到那些犯罪率相对较低的州，

例如新罕布什尔州（New Hampshire）和罗德岛州（Rhode Island）。而还有一些州，例如北卡罗来纳（North Carolina），只有某单一类型的罪案发生率较高。

对于这套数据来说，我对这种格式已经比较满意了。不过它还有两种变体，也许大家会希望尝试。第一种将所有数据都限制在了圆形的上半部分，如图7-17所示。

```
stars(crime, flip.labels=FALSE, key.loc = c(15, 1.5), full=FALSE)
```

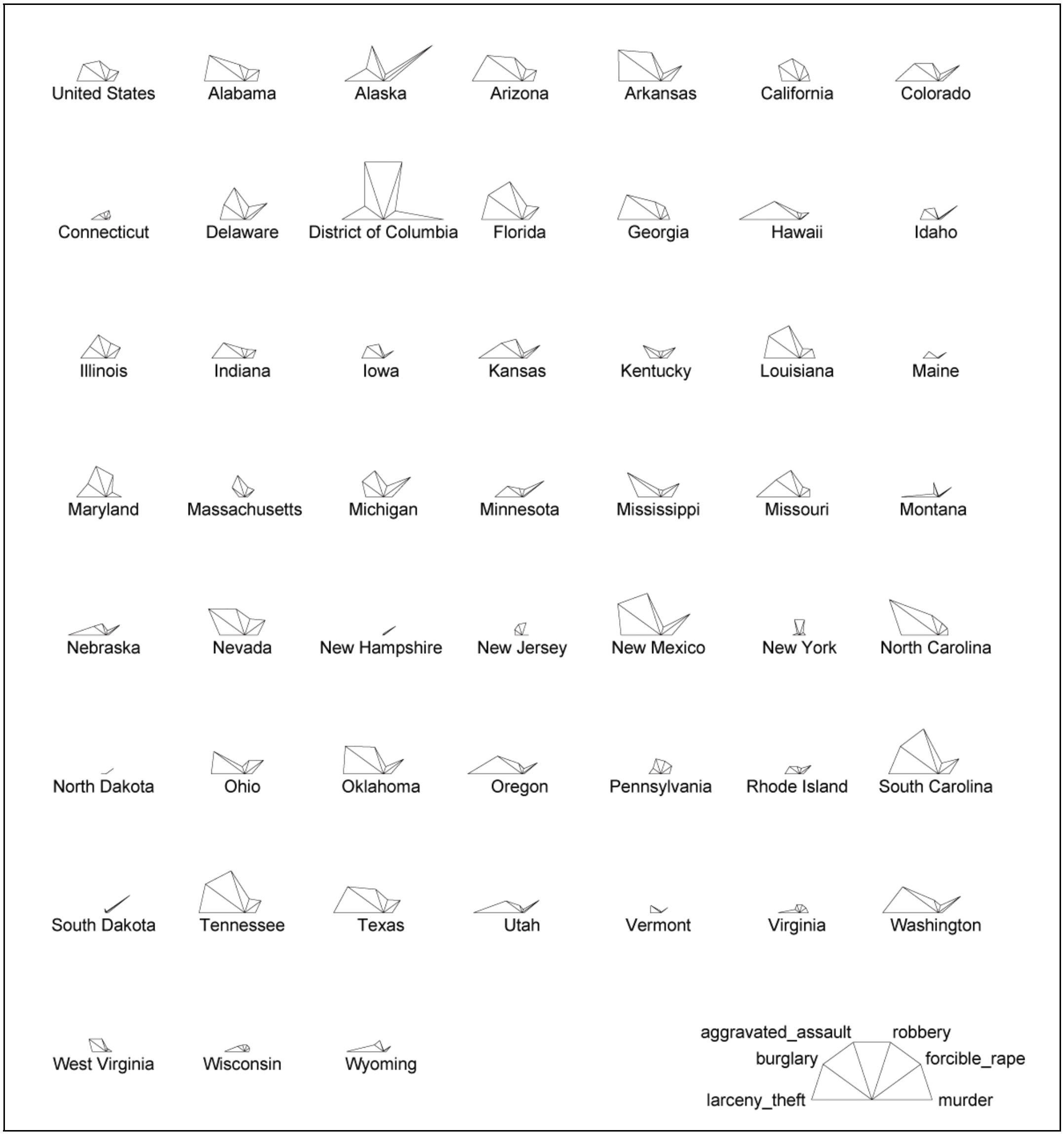


图7-17　将各轴限制在上半圆的星图

第二种变体则体现了各个扇形的长度，而不只是各轴终点的位置，如图7-18所示。这种图表其实已经不是星图了，而是南丁格尔图（也叫极坐标区图，polar area diagram），不过大家明白意思就行。如果你选择这种表现形式，可能需要尝试使用不同的颜色主题，而不是默认的这种怪异颜色。

```
stars(crime, flip.labels=FALSE, key.loc = c(15, 1.5), draw.segments=TRUE)
```

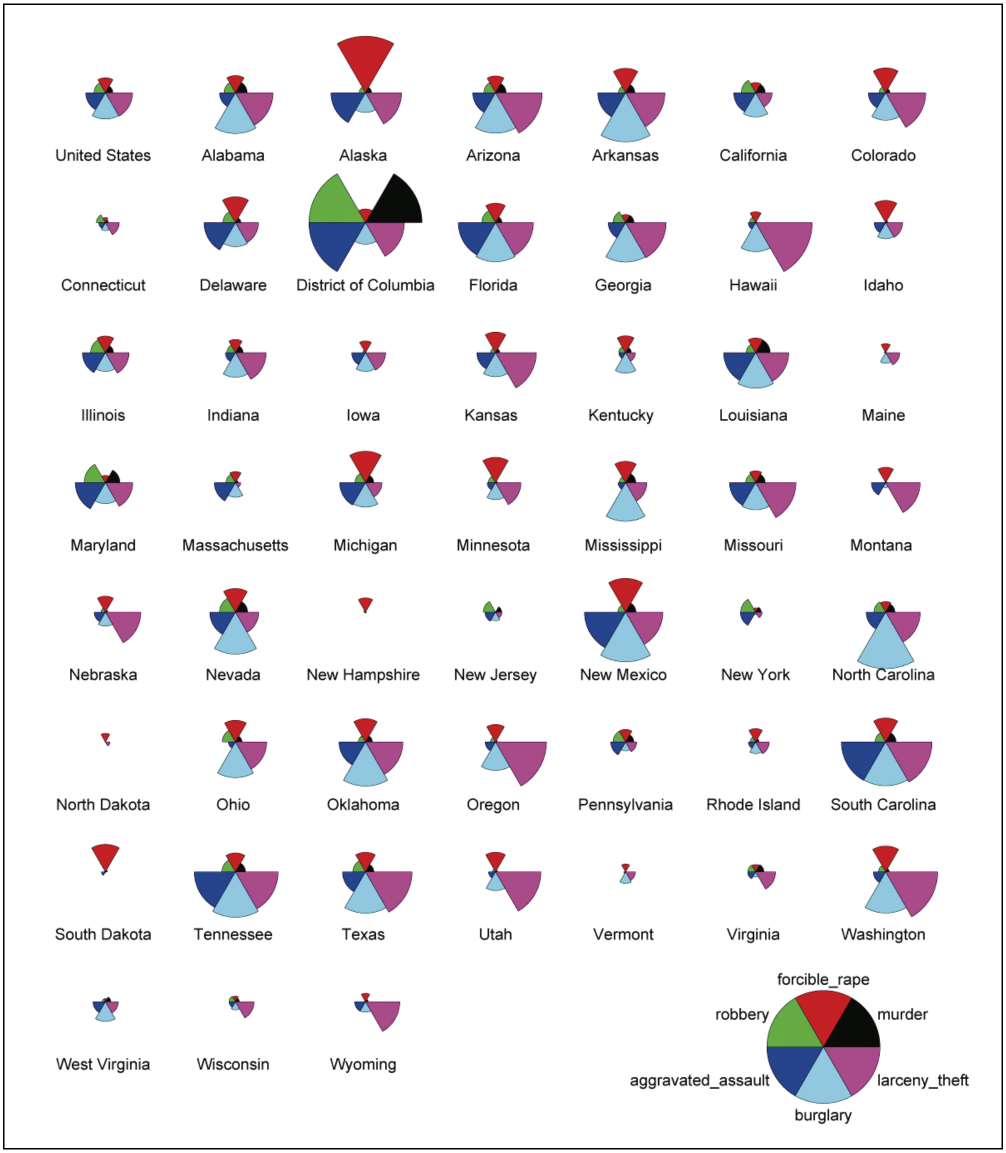


图7-18 南丁格尔图显示的犯罪率（另见彩插图7-18）

就像我刚才所说，我对图7-16所示的那种原始格式已经很满意了，所以可以把它导入到Illustrator里面做一些优化。这幅图表并不需要太多修改工作。在各行之间略微增加空白区域可以减少标签引起的歧义，另外可以把说明挪到顶端，让读者一开始就知道要阅读的内容（见图7-19）。其他的就没什么了。

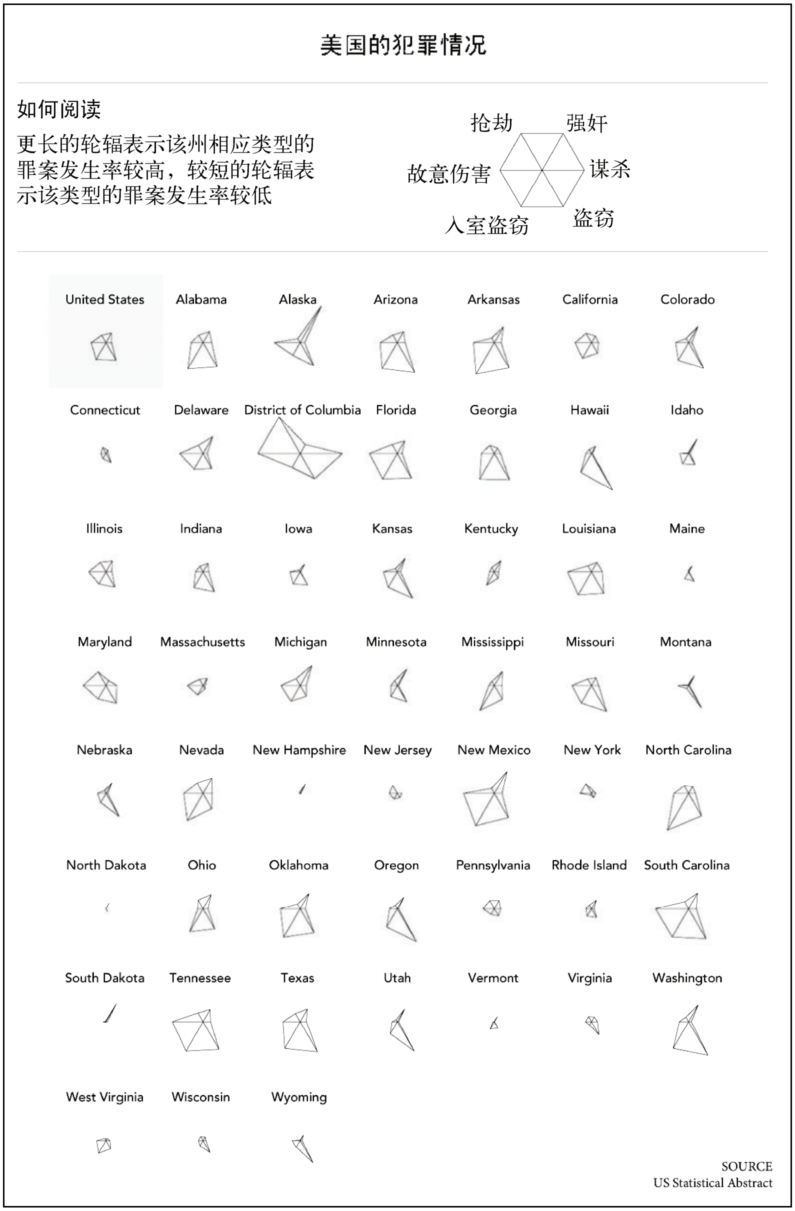


图7-19　显示各州犯罪率的系列星图

7.2.4　平行前进

尽管星图和切尔诺夫脸谱图能方便地找出各个对象与同类之间的差异，但它们却很难描述群组或各变量之间的关系。1885年由Maurice d’Ocagne发明的平行坐标图能帮我们解决这个问题。

如图7-20所示，我们将多条轴平行放置。每条轴的顶端表示该变量的最大值，底端表示最小值。对于每个对象来说，会出现一条由左到右的线条，根据各个变量数值的不同而上下浮动。

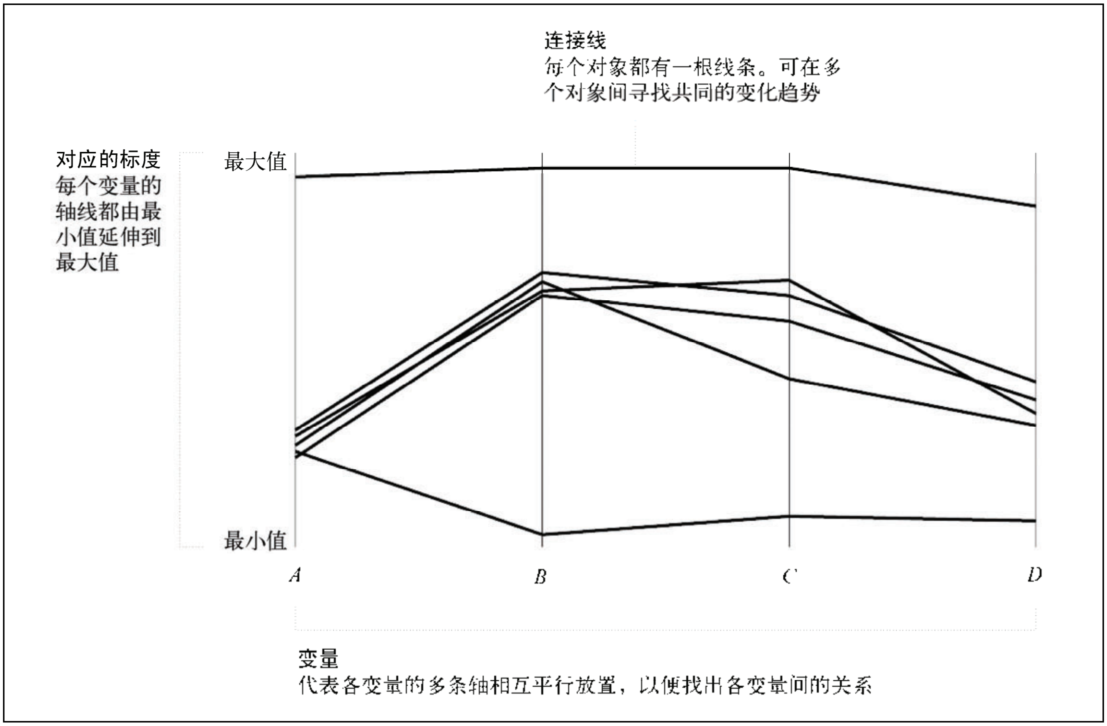


图7-20　平行坐标图的基本框架

假设我们用之前的篮球数据来绘制这种图表。为了简单起见，我们只按顺序绘制得分、篮板和犯规数量。如果有一位球员是顶级得分手，但抢篮板能力很弱，又喜欢犯规，那么在平行坐标图中，该球员的线条就会一开始很高，然后降低，然后再次升高。

当我们绘制更多对象时，这一方法就能帮助我们找出群组和趋势的变化。在下面的例子中，我们将用平行坐标图表现来自美国国家教育统计中心的数据。

创建平行坐标图

从可交互方面来看，绘制平行坐标图有多种选择。如果想创建自定义图形，我们可以在Protovis里面构建，或者把数据载入到诸如GGobi这样的实验性工具中。这些方法可以让我们过滤出并高亮显示自己感兴趣的数据点。不过，我还是倾向于绘制静态的平行坐标图，因为这样可以一次性比较各种不同的过滤结果。如果是可交互版本，我们就只能得到一张图，而且如果在一个地方显示一大堆高亮的结果，会很难看清内容。

> ▶ 访问http://ggobi.org免费下载GGobi。

第一步想必大家都知道了。在开始进行可视化之前，我们需要数据。在R中用`read.csv()`来载入我们的教育数据。

```
education <- read.csv("http://datasets.flowingdata.com/education.csv",
header=TRUE)
education[1:10,]
```

数据共有7列。第一列是各州的名称，其中“United States”是全国平均数据。其后3列依次是各州高中生在阅读、数学和写作各科目的SAT[①]平均得分。第5列是各州高中毕业生参加SAT考试的百分比，最后2列则是各州的高中毕业生就业率和高中生辍学率。我们感兴趣的是这些变量是否相互关联，以及是否存在任何明确的分组。比如说，那些高辍学率的州是否SAT的平均得分也会较低？

R本身并不支持直接生成平行坐标图，但lattice工具包可以。我们先载入该工具包（如果没有该工具包，请先安装）。

```
library(lattice)
```

很好，现在就简单了。lattice工具包提供了一个`parallel()`函数，可以直接使用。

```
parallel(education)
```

这将生成如图7-21所示的图表。很好，但这实际上没什么用。在图上有太多的线条，而且变量是从上到下放置的，不是从左到右。现在这张图看上去就像是彩虹色的意大利面。

要怎样来修改这幅平行坐标图才能从中获得信息呢？首先，我们应该将它横过来放置。当然这并不是一条硬性规则，只是个人喜好问题，但从左到右的平行坐标看上去明显更符合习惯，如图7-22所示。

```
parallel(education, horizontal.axis=FALSE)
```

① SAT曾先后指学术能力考试（Scholastic Aptitude Test）和学术评估测试（Scholastic Assessment Test），但现在并无确切的含义。它可以说是美国高中生的“高考”，但高中生能否就读四年制大学，并不仅仅取决于SAT的分数高低。

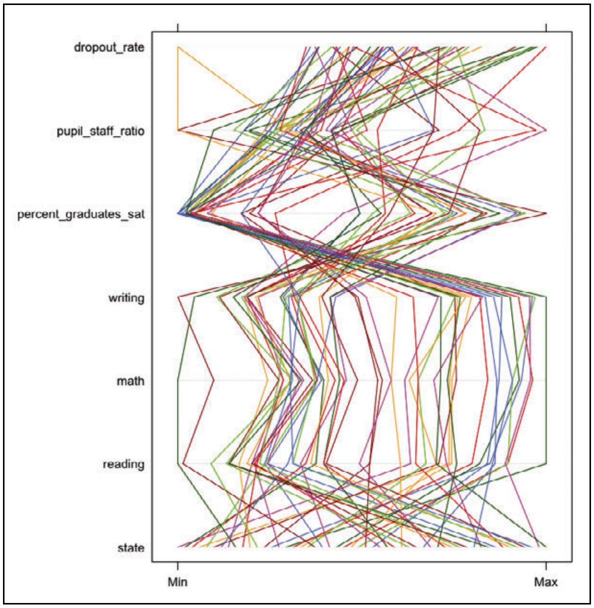


图7-21　用lattice工具包生成的默认平行坐标图

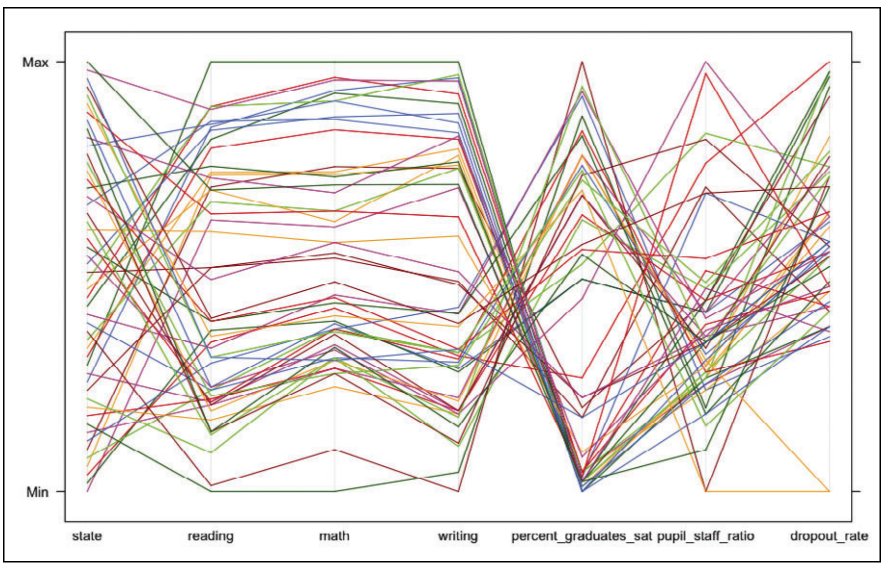


图7-22　水平放置的平行坐标

我们也不需要州名那一列，这是因为：第一、州名并不是数值；第二、每个州的名字都不相同。现在把线条的颜色都改为黑色——我很喜欢颜色，但现在也太多了点。执行以下代码，我们就会得到图7-23。

```
parallel(education[,2:7], horizontal.axis=FALSE, col="#000000")
```

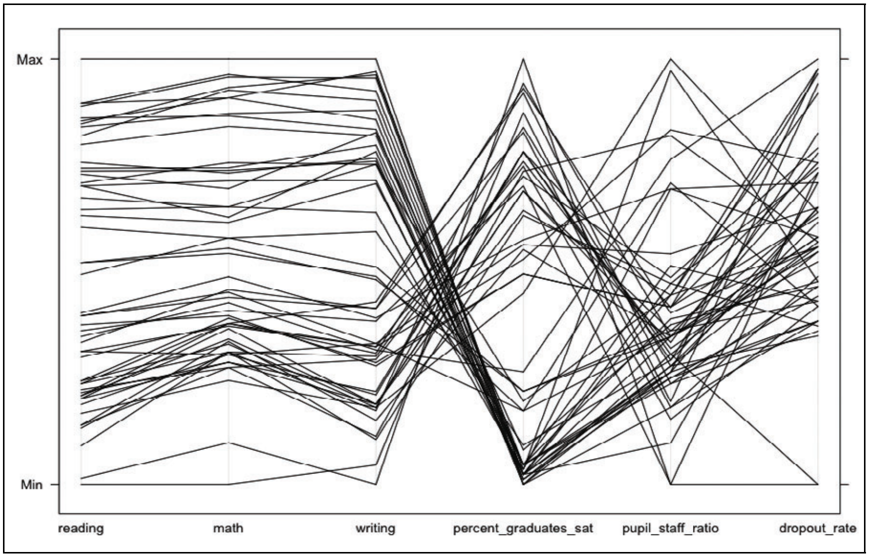


图7-23 简化后的平行坐标

现在看上去好些了。从阅读、数学到写作科目的线条很少相互交错，几乎是平行前进的。这表示那些阅读成绩好的州，同样数学和写作的成绩也会较好。与之类似，阅读成绩较差的州在数学和写作方面也不尽如人意。

从SAT成绩继续往右，在各州的SAT考试参加率这里，有趣的事情发生了。看上去似乎在那些SAT平均得分较高的州，参加SAT考试的学生却并不占多数。而SAT平均得分较低的州情况却正相反。我对教育领域并不算太熟悉，但我猜这是因为有些州里每个高中毕业生都参加了SAT考试，而在其他州里，如果学生不想上大学，他们就不会参加SAT考试。所以，如果要求那些对考试结果并不关心的学生去参加考试，平均分自然就会被拉低了。

我们可以加入一些对比色，让这一点体现得更加明显。在`parallel()`函数中，可以通过`col`参数来完全控制颜色。在之前，我们只用了一种颜色（#000000），但也可以向它传递一个颜色数组，给每一行数据设定一个颜色值。现在让我们将阅读得分排名前50%的州设置为黑色，将它们下面的那些州设置为灰色。首先用`summary()`来找到教育数据中阅读科目得分的中位数。在R中输入`summary(education)`，它会返回所有列的状态摘要，我们很容易就能发现，阅读（reading）科目得分的中位数是523。

```
        state        reading            math          writing
Alabama    : 1   Min.   :466.0   Min.   :451.0   Min.   :455.0
Alaska     : 1   1st Qu.:497.8   1st Qu.:505.8   1st Qu.:490.0
Arizona    : 1   Median :523.0   Median :525.5   Median :510.0
Arkansas   : 1   Mean   :533.8   Mean   :538.4   Mean   :520.8
California: 1   3rd Qu.:571.2   3rd Qu.:571.2   3rd Qu.:557.5
```

```
Colorado  : 1   Max.   :610.0   Max.   :615.0   Max.   :588.0
(Other)   :46
percent_graduates_sat pupil_staff_ratio  dropout_rate
Min.   : 3.00         Min.   : 4.900    Min.   :-1.000
1st Qu.: 6.75         1st Qu.: 6.800    1st Qu.: 2.950
Median :34.00         Median : 7.400    Median : 3.950
Mean   :37.35         Mean   : 7.729    Mean   : 4.079
3rd Qu.:66.25         3rd Qu.: 8.150    3rd Qu.: 5.300
Max.   :90.00         Max.   :12.100    Max.   : 7.600
```

现在遍历数据的每一行，检查它是大于还是小于该中位数，并指定相对应的颜色。c()指令创建了一个空的向量，每次循环都会往其中添加一个值。

```
reading_colors <- c()
for (i in 1:length(education$state)) {

    if (education$reading[i] > 523) {
        col <- "#000000"
    } else {
        col <- "#cccccc"
    }
    reading_colors <- c(reading_colors, col)
}
```

然后将reading_colors数组传递到parallel()里面，而不是一个单独的"#000000"。这会让我们得到图7-24，从中可以很容易地看出由高到低的大转移。

```
parallel(education[,2:7], horizontal.axis=FALSE, col=reading_colors)
```

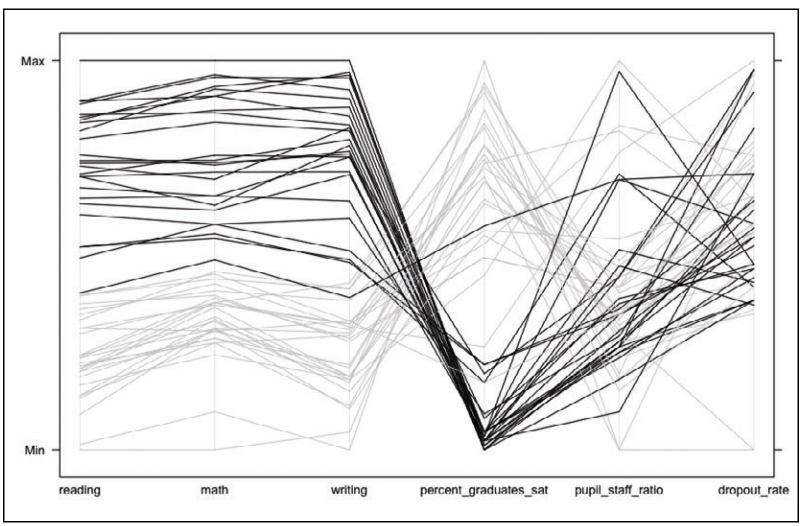


图7-24 阅读成绩较好的州被高亮显示

7

那么在辍学率方面，情况又是如何呢？我们可以对辍学率进行与阅读得分相同的处理，只不过这次用第三个四分位数，而非中位数。这个四分位数是5.3%。再次，我们遍历每一行数据，只不过这次检查的是辍学率，而非阅读科目的平均得分。

```
dropout_colors <- c()
for (i in 1:length(education$state)) {

    if (education$dropout_rate[i] > 5.3) {
        col <- "#000000"
    } else {
        col <- "#cccccc"
    }
    dropout_colors <- c(dropout_colors, col)
}
parallel(education[,2:7], horizontal.axis=FALSE, col=dropout_colors)
```

图7-25显示了得到的结果，这次的效果不如前一个图表那样引人注目。从视觉上来看，在所有变量中并没有明显的分组趋向。

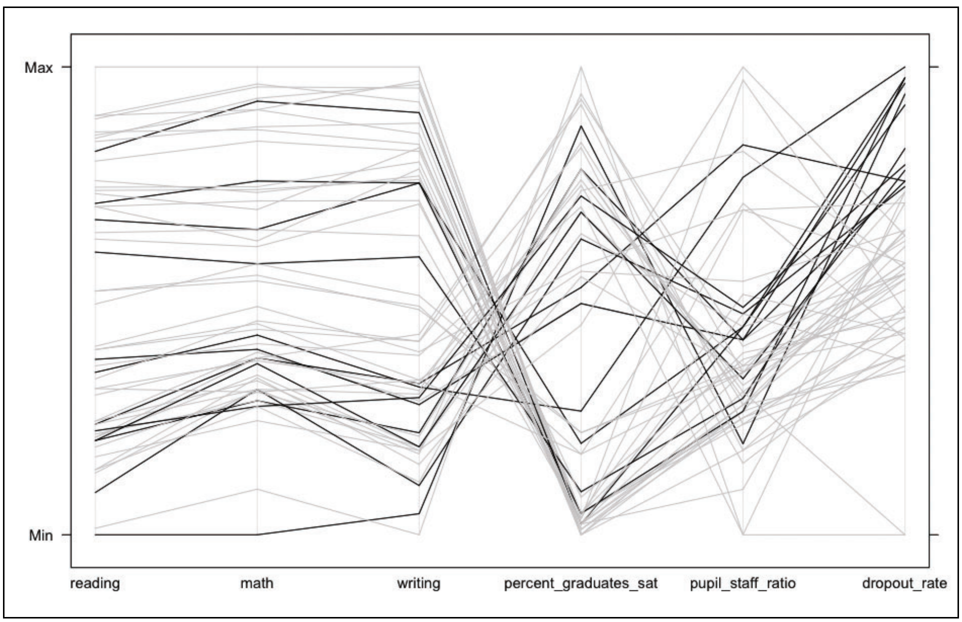


图7-25　辍学率较高的州被高亮显示

大家也可以在自己感兴趣的方面进行更多的探索。现在让我们回到图7-24并进行优化。美化标签可以让它们更加明显。或者加入一些颜色来代替单纯的灰度？或者加入一段简短的说明，以解释为什么50%的州都被高亮显示？最终得到的结果将如图7-26所示。

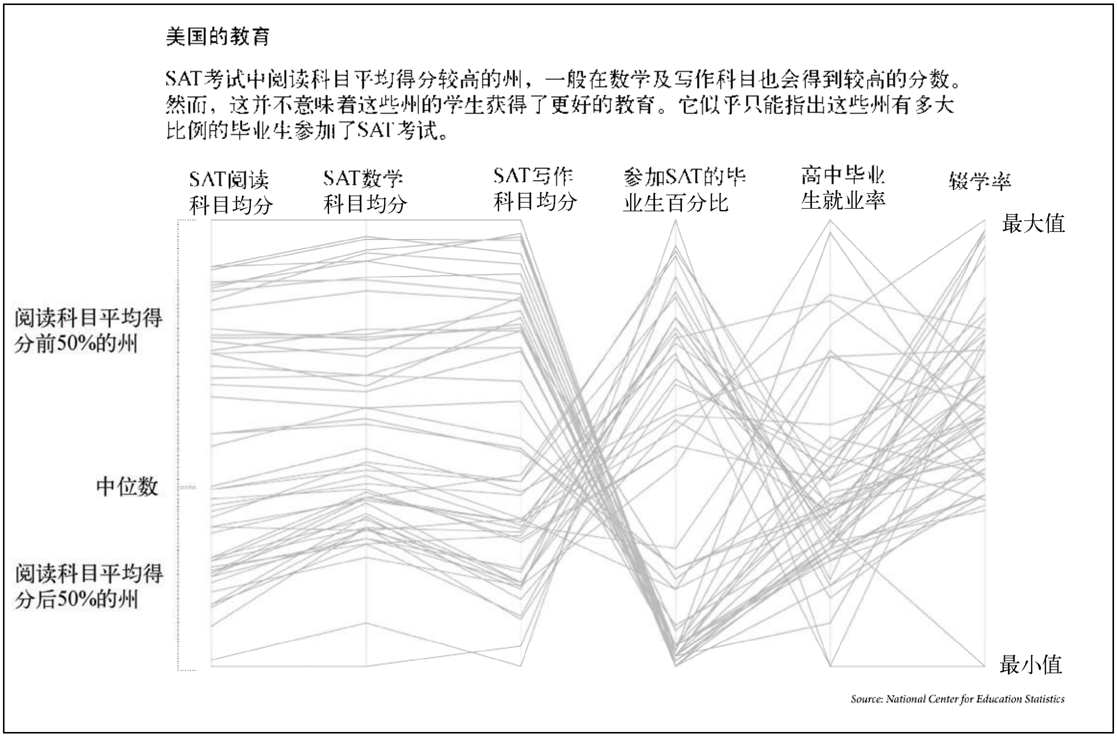


图7-26　关于SAT得分的平行坐标图表（另见彩插图7-26）

7.3　减少维度

当我们使用切尔诺夫脸谱图或平行坐标图时，主要的目的是去减少。我们希望在数据集或者全体中找出不同的分组。这里的挑战在于，我们并不总是清楚从哪里开始观察这些脸谱或者连接线，所以如果能根据某些标准将对象划分为不同的群集，事情就会容易得多。这就是多维量法（MultiDimensional Scaling，MDS）的目的之一。将所有事物都考虑进来，然后在图表上将相互更类似的对象靠近放置。

这一主题内容极为丰富，专门介绍这一主题的书籍就有很多，因此要具体解释起来恐怕会过于技术化。为了简单起见，我会只停留在概念层面，把背后的算法暂时放在一边。不过话又说回来，多维量法是我在读研时学到的第一个概念，如果你希望深入了解统计学的话，它背后的理论的确值得花时间好好学习。

> ▶ 有关该方法的更多细节，请查阅多维量法或主成分分析。

假设你现在和另两个人在一个正方形的空房间内。你的任务是根据这两个人的身高安排他们

站在合适的位置。他们的身高越接近，就应该彼此靠得越近，如果身高相差越远，彼此的距离就应该拉得更大。其中一人很矮，而另一人很高。这两人应该怎么站？他们应该站在相对的角落里，因为他们的身高是截然相反的。

现在又进来了一个人，他的身高中等。根据我们的安排规则，这个新来的应该站在房间的中心，就在之前两个人的正中间。他与高个子的身高差距和矮个子是一样的，因此他和两个人的距离彼此相等。与此同时，高个子和矮个子之间依然保持最大距离。

好，现在再引入另一个变量：体重。你知道所有这三个人的身高和体重。矮个子和中等个子的体重完全相等，而高个子的体重最轻。那么，根据身高和体重，你如何安排这三个人在房间中的位置？如果保持前两个人（矮个子和高个子）的对角位置，那么第三个人（中等个子）就应该向矮个子靠近，因为他们的体重是相同的。

现在你理解我们的安排规则了吗？两个人越相似，他们彼此间的距离就应该越靠近。在这个简单的例子里，你只有3个人和2个变量，很容易人工计算出结果来。但如果有50个人，而且你需要根据5种标准来安排他们在房间里的位置，就会棘手得多。而这就是多维量法的用武之地。

利用多维量法

用实例来理解多维量法会容易得多，所以让我们再次使用之前的教育数据来看看。首先还是在R中载入数据。

```
education <-
    read.csv("http://datasets.flowingdata.com/education.csv",
    header=TRUE)
```

别忘了，每个州都有一行，其中包括美国全国平均值和哥伦比亚特区。每个州都有6个变量：阅读、数学和写作科目的SAT成绩，参加SAT考试的毕业生百分比，高中毕业生就业率，以及辍学率。

和刚才那个房间的比喻类似，只不过这次不再是正方形的房间，而是一个图表；不再是一个个的人，而是一个个的州；不再是身高和体重，而是和教育相关的各项指标。我们的目的还是一样：在一个*x*-*y*坐标的图表里面放置各个州，使指标相似的州之间的距离比较接近。

第一步：计算出每个州和其他各州之间的距离。用`dist()`函数可以做到这一点。我们只需要第2列~第7列，因为第1列是各个州的名称，而且我们都知道它们是不同的。

```
ed.dis <- dist(education[,2:7])
```

在R的控制台中输入`ed.dis`，我们会看到一系列矩阵。每个单元格表现出该州和另一个州之间相距多远（欧几里得像素距离）。比如说，第2行第2列的值就是阿拉巴马州和阿拉斯加州之间的距离。对象在此时并不太重要，重要的是相互之间的差距。

怎样将这个51 × 51的矩阵变成一幅*x*–*y*轴上的图表呢？现在还不行，除非我们为每个州建立一套*x*-*y*轴的坐标。这就是`cmdscale()`函数的作用。它会把距离矩阵作为输入，然后返回一系列点，这些点之间的差距将和矩阵所指定的保持一致。

```
ed.mds <- cmdscale(ed.dis)
```

再次在R的控制台中输入`ed.mds`，我们就能看到现在每一行数据都有自己的*x*–*y*轴坐标了。将这些值存储为变量`x`和`y`，然后放到`plot()`函数里面，就会看到结果（见图7-27）。

```
x <- ed.mds[,1]
y <- ed.mds[,2]
plot(x,y)
```

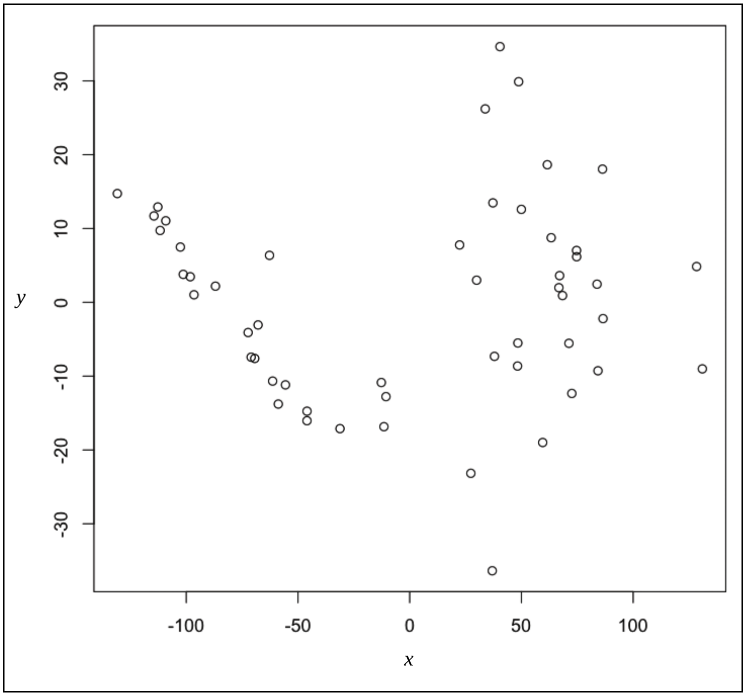


图7-27 显示多维量法结果的点状图

看起来不错。每个圆点都代表一个州。不过还有一个问题：我们不知道哪个点代表哪个州。我们还需要标签，所以跟之前一样，用`text()`把州名放在各圆点的位置上，结果如图7-28所示。

```
plot(x, y, type="n")
text(x, y, labels=education$state)
```

相当酷。我们能看到图中出现了两个集群，一个在左边，一个在右边。美国（United States）则位于右边集群的底部，略靠中间的位置。到了这一步，你可以自行决定是否继续分析各个集群分别代表什么意思，不过要想开始你的数据探索游戏，这是一个好的起点。

比如说，你可以像之前在平行坐标图里面一样，用`dropout_colors`来为各个州着色，如图7-29所示。它并不能告诉你太多信息，但确实验证了之前你在图7-25里面看到的内容。

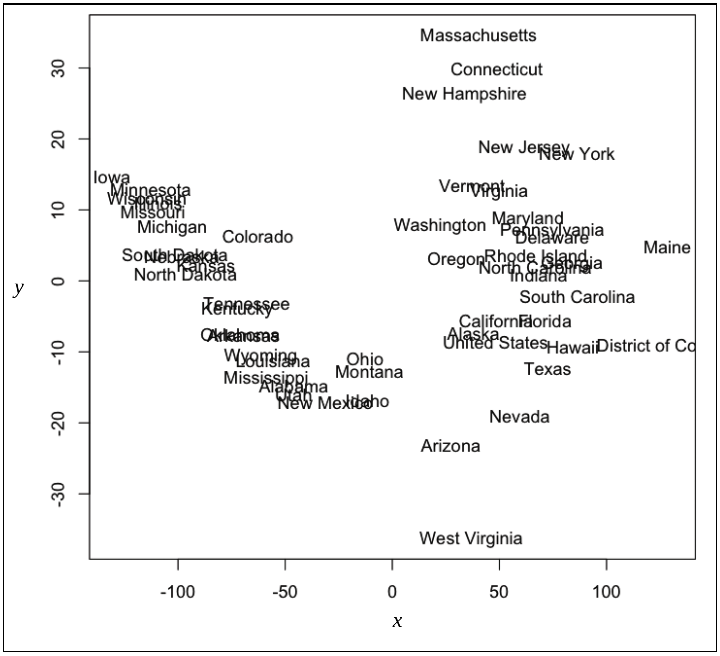


图7-28　用州名来代替圆点，以便观察每个州被放在了什么位置

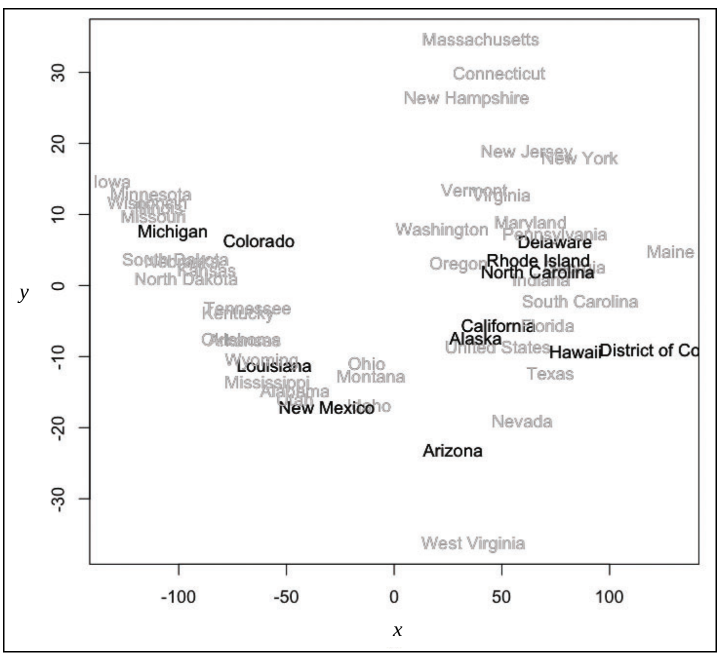


图7-29　按辍学率进行着色的各个州

如果按阅读科目的平均分来着色，结果又会怎样？你当然可以这么做，如图7-30所示。啊，现在似乎已经浮现出清晰的模式了。高分的州都在左边，低分都在右边？是什么让华盛顿州与众不同？仔细看看——稍候你就知道答案了。

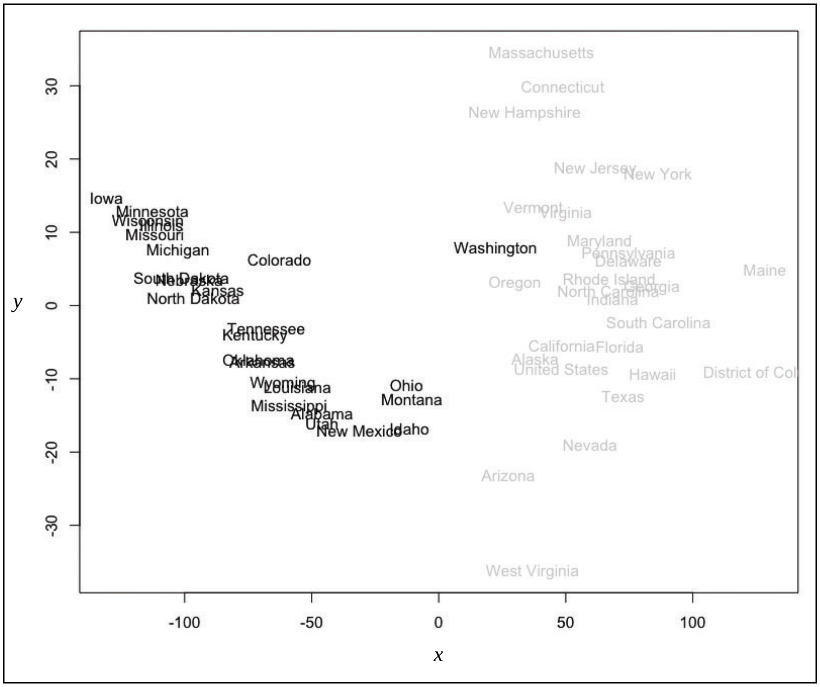


图7-30　按阅读科目平均分进行着色的各个州

如果你还想玩点花样，可以尝试一种被称为“基于模型聚类”（model-based clustering）的方法。我不会详细讨论它的细节，只是告诉大家如何实现，并且证明这些并不是魔术，只会涉及数学运算。简要来说，这种方法就是在你的多维量法图表中用`mclust`工具包来标识出各个集群。如果你还没有`mclust`工具包，可以安装一个，然后运行下列代码，将得到如图7-31所示的图表。


```
library(mclust)
ed.mclust <- Mclust(ed.mds)
plot(ed.mclust, data=ed.mds)
```

左上角的那幅图表显示了运行算法在数据中找到理想集群数量的结果。其他三幅图表则显示了集群。非常厉害。我们得到了两个集群，现在则更加明确了，显示了高和低的状态。

一般到这个时候，我就该告诉大家将PDF文件导入到Illustrator中进行优化了。不过我不确定自己是否会将这些图表发表给一般读者。对于非专业的读者来说，这些图表过于抽象，很难理解它们的含义。它们对于数据探索来说非常有价值，但我建议还是多使用标准的设计原则为好。找出清楚讲出故事所需要的素材，然后抛掉其他那些。

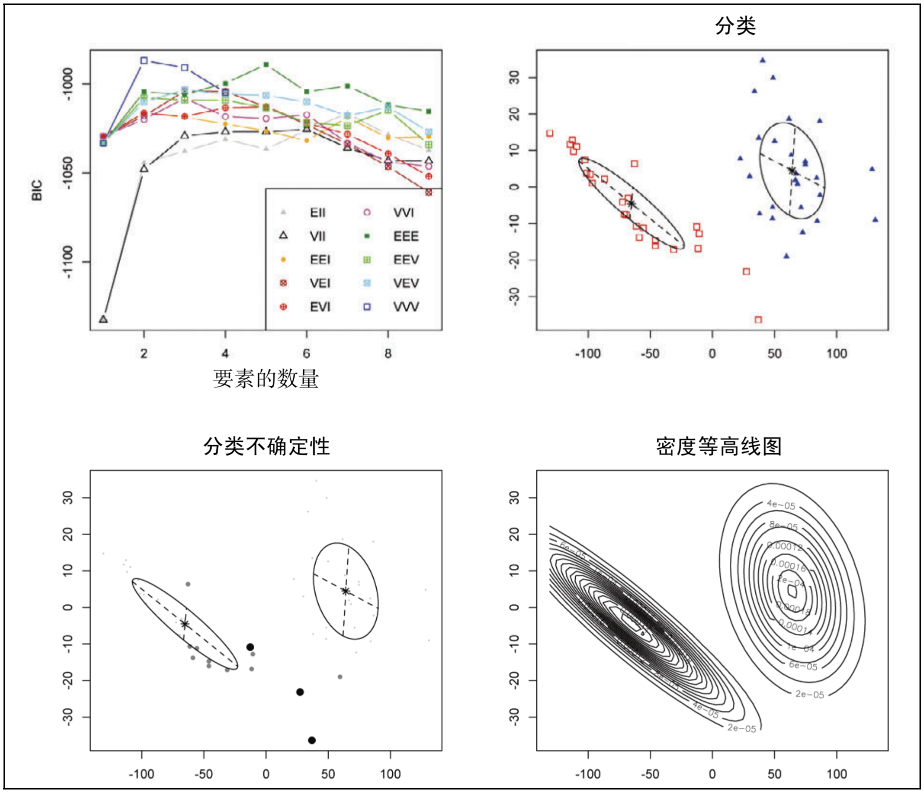


图7-31　基于模型聚类得到的结果

7.4　寻找异常值

一方面，我们要探究数据对象为什么属于某个群组，另一方面，也应该探究它们为什么会不属于某个群组。也就是说，总会有一些数据点从同类中凸显出来，正如你所猜到的，它们被称作为“异常值”（outlier）。这些数据点和全体中的其他数据显得格格不入。有时候它们正是你的故事中最值得人注意的亮点，有时候它们可能只是无聊的笔误，比如少输了一个零。不管怎样，我们都需要对它们进行检查，以便了解到底发生了些什么。谁都不希望在绘制了一幅巨大的图表后，被细心的读者指出某个异常值有问题，导致之前的辛苦工作付诸东流。

有很多人设计了专门的图表类型，方便强调显示异常值。不过以我的经验来看，什么都比不上基础图表和常识来得有效。深入理解数据的上下文背景，做好功课，如果对某些问题不太确定，不妨询问该方面的专家。找到异常值后，我们完全可以用之前所掌握的图表绘制技巧对它们进行强调：添加不同的颜色、使用箭头或者更粗的边框等。

现在来看一个简单的例子。图7-32是一个时间序列图，显示了从Weather Underground网站上抓取的1980—2005年的天气数据（和我们在第2章中所做的一样）。图表中按季节的循环是在我们

预料之中的，不过中间那一段是什么情况？看上去不寻常的平滑，而其他地方都有很多噪点。这没有什么大不了的，但如果你碰巧需要依靠这些数据建立天气模型，就需要了解哪些是真实数据，哪些又是估算出来的。

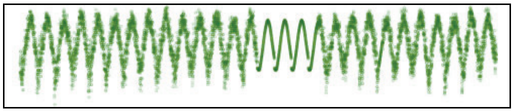


图7-32 Weather Underground网站估算的天气数据

与之类似，我们再看看之前显示犯罪率的星图，很明显哥伦比亚特区过于突出。用基本的柱形图也能轻易发现这一问题，如图7-33所示。拿这些州与更像一座城市的哥伦比亚特区进行比较，是否有失公平？你可以自己来判断。

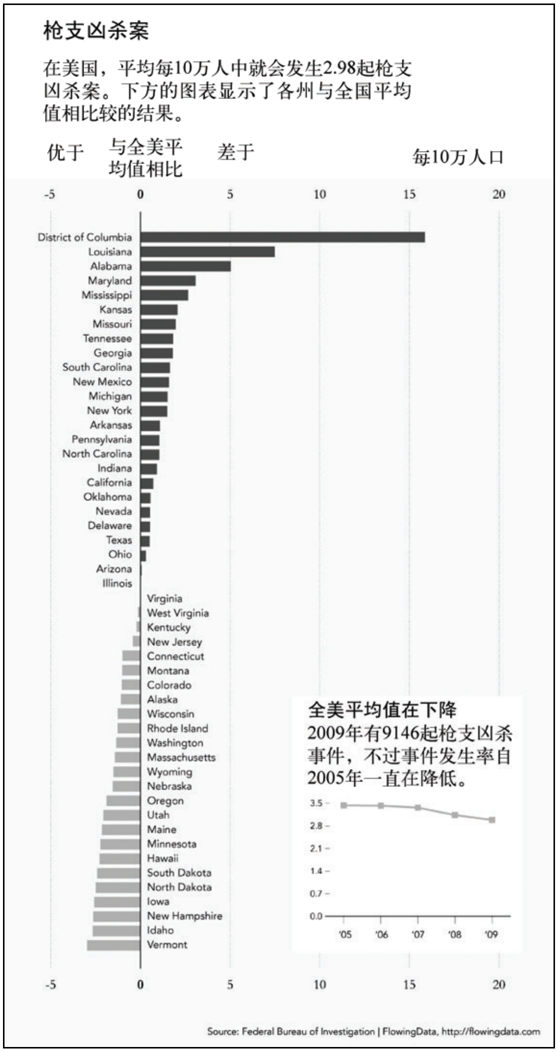


图7-33 美国的枪支凶杀案

而在第4章中谈到的订阅者数量问题又是怎么回事？在图7-34中，中间有一个明显的波谷，似乎FlowingData网站有一半以上的读者都蒸发了。

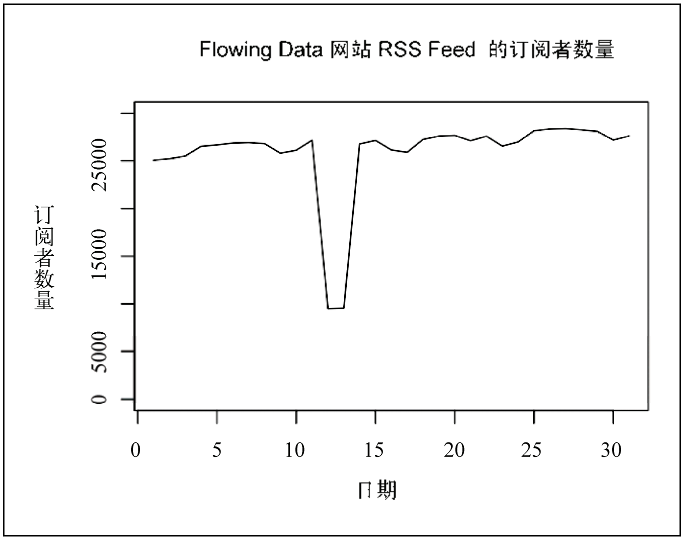


图7-34　FlowingData一段时期内的订阅者数量

我们也可以通过直方图来看看整体的分布情况，如图7-35所示。大部分订阅者都在右边，有一小部分在最左边，而中间什么都没有。

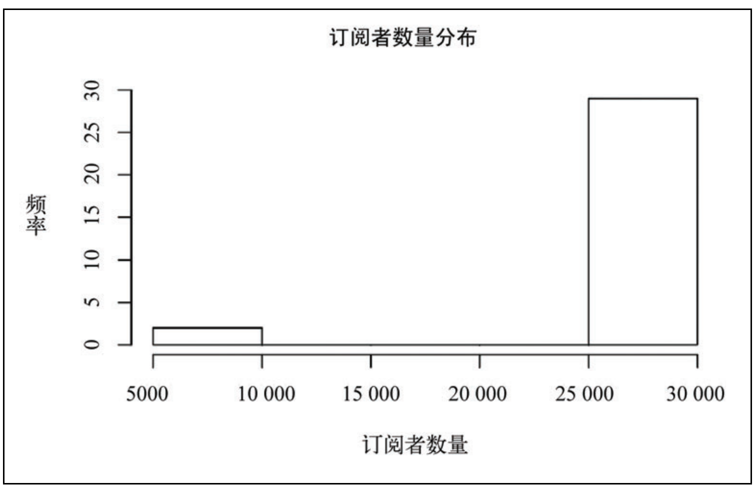


图7-35　显示订阅者数量分布的直方图

要想更具体一些，我们还可以用到箱型图（boxplot），它能显示分布中的四分位数。箱型图在R里面可以通过`boxplot()`函数生成，它会自动高亮显示比上四分位数、下四分位数多或少1.5倍以上的数据点（见图7-36）。

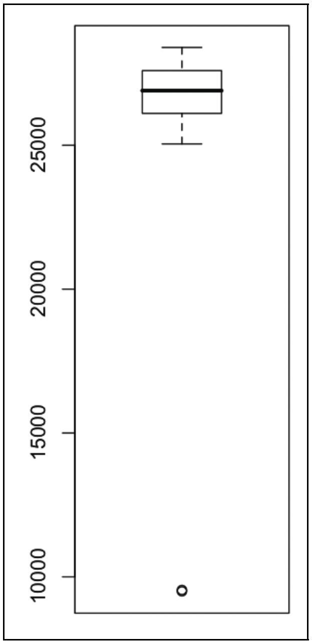


图7-36 显示订阅者数量分布的箱型图

> ▶ 一个数据集中有3个四分点（quartile），这3个四分点将图表4等分。中间的四分点也就是中位点（median），上四分点以上是所有数据中最大的25%，而下四分点以下则是所有数据中最小的25%。

如果我网站的订阅者数量不多，比如只有个位数，那么这么高百分比的降低还是可能的。但如果说我在网站里说了什么直接导致上万名读者变节（而且在几天后又都回来了），这似乎过于牵强。更合理的解释应该是Feedburner（我使用的feed发送服务商）给我的报告中出现了错误。

这些数据集中的异常值之所以如此显而易见，是因为我们对这些数据还算熟悉。要是我们用到了不太熟悉的数据集，那么它们可能就不会这么明显。如果发生了这种情况，不妨直接寻访数据的源头，问问是谁在负责此事。创建、存储这些数据的个人或团体通常都会很高兴有人在使用他们的数据，也会乐于提供建议。如果你确实无法找出更多细节，起码你也尽力了，那么就在图表的里面对问题给出一点说明吧。

7.5　小结

对于初学者来说，设计数据图表时最困难的一个环节就是找到入手点。所有的数据都摆在面前，但对它们都代表什么没有任何的概念或预期。一般来说，我们可以尝试对数据提出一个或多个疑问，并试图解决这些疑问，以此作为开始。但如果不知道要问什么又该怎么办？本章所讲述的方法能为我们带来很大帮助。它们能让我们一次性看到所有的数据，从而更方便地弄清楚下一步该探索其中的哪个部分。

不过，这只是个开始。我们在这一步可以将范围缩小到那些令人感兴趣的点上面。有了这些点，再加上前面几章中所讲的内容，应该可以帮助你对各种类型的数据进行深入的挖掘了。唔，除了一种数据之外。下一章就会谈到这种特殊的数据类型：空间数据。准备好画地图吧。

8 有关空间关系的可视化

地图是一种非常直观的可视化类型。我在还是个孩子的时候就能看懂地图。我还记得当时坐在父亲车里的副驾驶位置上，打开巨大的地图大声为他指路。如今这个角色已经变成了一位澳大利亚女士，从仪表盘的一个小盒子里用平缓而机械的声音迸出应该前进的方向。

不论怎样，地图都是一种理解数据的极佳手段。它们是真实世界按比例缩小后的版本，而且它们无处不在。在本章中，我们会深入到多种空间数据集中，寻找跨越空间和时间的那些模式。我们会先用R创建一些基础地图，然后用Python和SVG创建更高级的地图，最后，用ActionScript和Flash来创建可交互的动画地图。

8.1　在空间中寻求什么

我们阅读地图的方式和阅读静态图表的方式几乎是一样的。当我们在地图上寻找某个具体地点时，实际上仍然是一个不断缩小区域、或者不断和其他地区进行比较的过程。区别在于此刻我们面对的是经度和纬度，而非*x*轴与*y*轴坐标。在地图上，坐标彼此之间的联系和城市之间的联系也是一样的。*A*点和*B*点之间的距离为具体的英里数，而且从*A*到*B*所需要的时间也可以估算出来。而相对地，在点状图中*A*和*B*之间的距离则是抽象的，而且（通常）没有单位。

这种区别为地图和地图的制作带来了许多微妙之处。因此不难理解为什么《纽约时报》的图形编辑部会成立一个小组来专职设计地图。我们需要确保所有地点的位置都正确无误、颜色都符合规范、标签不会盖住位置，以及使用了正确的投影法。

本章只介绍了其中一小部分基础知识。仅仅这些就能够带领你在数据中挖掘出精彩的故事，但别忘了还能尝试许多其他超级炫的效果。

当引入时间之后，事情还会变得更加有趣。一幅地图只能代表一个时间片段，但我们可以用多幅地图来代表多个时间片段。我们甚至还可以用动画的形式来表现变化，例如某个企业在某个地理区域中的扩张（或萎缩），这样一来在某些地区的爆发性增长将会变得非常显而易见。另外，如果地图是可交互的，读者就能更方便地观察自己感兴趣的地区是如何发生变化的。在柱形图或点状图中我们不太可能得到这样的效果，但有了地图，数据立刻就能让人专享了。

8.2　具体位置

一份地点清单是我们能得到的最好处理的空间数据了。我们有了一堆地点的经纬度，只需将它们标注在地图上即可。也许你还想展现事件（比如犯罪）在哪些地方发生，或者想找到事件发生最密集的是哪个区域，这些都很容易办到，而且有很多方法。

在网上，调用地图最常用的方法是借助Google或者Microsoft地图。通过他们的地图API，我们就能立刻得到可以缩放和平移的可交互地图，而这只需要几行JavaScript代码就能实现。对于如何运用这些API，有海量的在线教程，而且说明文档也很详细，所以这里我就不再赘述了。

说明　Google和Microsoft为刚开始使用他们地图API的新手们提供了超级浅显易懂的教程，如果你打算用他们的产品实现一些基本的地图功能，不妨先过一遍这些教程。

不过这种地图也有不足。它们的可定制程度很低，到最后得到的结果看上去还是像Google或Microsoft地图。我不是说它们难看，只是当我们开发应用或设计图表时，地图在视觉上还是要与设计主题相吻合。有些办法能解决这个问题，但是为此花上太多精力又不值得，尤其是还有其他工具能做得更好的时候。

8.2.1 找到纬度和经度

在绘制地图之前，先考虑一下能拿到哪些数据，又需要哪些数据。如果你连需要的数据都没有，自然也就没法可视化了，不是吗？在大多数实际应用中，你需要经纬度才能确定地点的位置，而大部分数据集都不太可能会提供这些信息，我们能拿到的更有可能只是一份地址列表而已。

自然，你不能把街道名称和邮编直接硬插到地图里面，那样不可能会有好效果。我们必须首先拿到经纬度，而要想做到这一点，可以试试*地理编码*（geocode）。其原理是我们把地址传给服务提供商，服务提供商会请求数据库与地址进行匹配，然后判断这个地址应该是在地球的哪个位置，之后我们就能得到该位置的经纬度。

至于有哪些服务提供商可用，嗯，倒是有不少。如果需要地理编码的地点不多，我们可以去某个网站然后手工输入查询。Geocoder.us是一个很好的选择，而且免费。如果地点不需要特别精确，也可以试试Pierre Gorissen开发的基于Google地图的Latitude Longitude Popup。该应用有一个简单的Google地图界面，在地图上点击任何位置，它都会弹出该地点的经纬度信息。

但如果你有很多地点需要地理编码，就应该通过程序来解决了，否则手工的复制粘贴太花时间。Google、Yahoo!、Geocoder.us和Mediawiki都提供地理编码的API，而Python中有一个叫做Geopy的地理编码工具箱则把它们都囊括进来了。

有用的地理编码工具

- ❑ Geocoder.us（http://geocoder.us）——提供简单的界面，将地址复制并粘贴进去后会得到经纬度信息。还提供API。
- ❑ Latitude Longitude Popup（www.gorissen.info/Pierre/maps/）——Google地图的混搭模式。在地图上点击某个位置，它就能告诉你该地的经纬度。
- ❑ Geopy（http://code.google.com/p/geopy/）——Python的地理编码工具箱。在一个工具包中囊括了多个地理编码API。

访问Geopy的网页可以浏览该工具包的安装指南，还能找到很多简单的应用实例。我们下面的例子假设你已经在计算机上安装了该工具包。

在安装好Geopy后，在http://book.flowingdata.com/ch08/geocode/costcos-limited.csv下载位置数据。这是一个CSV文件，包含了美国境内每一家好市多（Costco）店铺的地址信息，但没有经纬度。这就得靠我们来想办法了。

打开一个新文件，将其存储为geocode-locations.py。和往常一样，首先导入你要用到的工具包。

```
from geopy import geocoders
import csv
```

我们还需要一个API开发密匙（API Key）才能获得各种服务。对于本例来说，我们只用从Google那里获取一个即可。

说明　访问http://code.google.com/apis/maps/signup.html注册一个Google地图API的免费开发密匙。过程非常简单，只需几分钟就能完成。

把开发密匙存入一个名为g_api_key的变量，在初始化地理编码时使用它。

```
g_api_key = 'INSERT_YOUR_API_KEY_HERE
g = geocoders.Google(g_api_key)
```

载入costcos-limited.csv数据文件，然后开始循环处理。针对每一行数据，我们把所有的地址信息拼成一个整体，然后进行地理编码。

```
costcos = csv.reader(open('costcos-limited.csv'), delimiter=',')
next(costcos) # 跳过标题

# 打印标题
print "Address,City,State,Zip Code,Latitude,Longitude"
for row in costcos:

    full_addy = row[1] + "," + row[2] + "," + row[3] + "," + row[4]
    place, (lat, lng) = list(g.geocode(full_addy, exactly_one=False))[0]
    print full_addy + "," + str(lat) + "," + str(lng)
```

这就行了。运行Python脚本，并把输出结果存储为costcos-geocoded.csv。以下是所得数据的头几行：

```
Address,City,State,Zip Code,Latitude,Longitude
1205 N. Memorial Parkway,Huntsville,Alabama,35801-5930,34.7430949,-86
.6009553
3650 Galleria Circle,Hoover,Alabama,35244-2346,33.377649,-86.81242
8251 Eastchase Parkway,Montgomery,Alabama,36117,32.363889,-86.150884
5225 Commercial Boulevard,Juneau,Alaska,99801-7210,58.3592,-134.483
330 West Dimond Blvd,Anchorage,Alaska,99515-1950,61.143266,-149.884217
...
```

相当酷。如果幸运的话，每一个地址都应该能找到对应的经纬度坐标。但这种情况通常不会出现。如果遇到了这种问题，你应该在之前代码的倒数第二行加入错误验证。

```
    try:
        place, (lat, lng) = list(g.geocode(full_addy, exactly_one=False))
[0]
        print full_addy + "," + str(lat) + "," + str(lng)
    except:
        print full_addy + ",NULL,NULL"
```

这段代码会尝试寻找对应的经纬度坐标，如果寻找失败，就会输出该行数据的地址以及空的坐标值。运行Python脚本并存储输出结果，我们就能从中找到那些空值。此时我们可以用Geopy试试其他服务商，或者在Geocoder.us里手工输入这些空值的地址进行查询。

8.2.2 单纯的点

现在我们有了带经纬度信息的各个地点，可以绘制地图了。最直接的办法就是为各个地点添加标记，和真实世界中往纸板上摁图钉很相似，只不过是让计算机来完成。基本框架如图8-1所示。

虽然这个概念很简单，但我们仍然能在数据中看到集群、扩展和异常值等特征。

1. 带有点的地图

虽然R在绘制地图方面的功能比较有限，但在地图上放置点却很轻松。这一任务主要是由`maps`工具包来完成的。在R中输入`install.packages()`，或者通过Package Installer安装`maps`工具包。安装好后将其载入到工作区。

```
library(maps)
```

下一步：载入数据。你可以使用之前地理编码过的好市多店铺的地址数据，或者也可以直接通过URL载入我加工好的数据集。

```
costcos <-
  read.csv("http://book.flowingdata.com/ch08/geocode/costcos-geocoded
  .csv", sep=",")
```

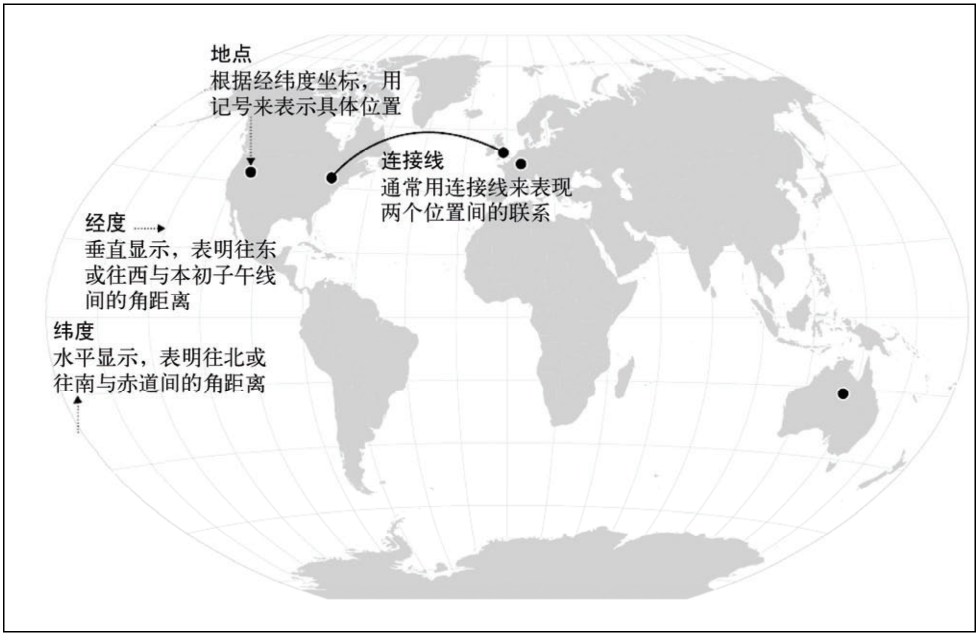


图8-1 地图标识地点的基本框架

现在开始绘图。在创建地图时，把它们看做是层会比较好理解（不管你用的是什么软件）。

最底层一般就是显示地域界限的基础地图，它的上面再是那些数据的层。在本例中，最底层是美国地图，而第二层就是好市多店铺的地址。以下代码可以创建第一层，结果如图8-2所示。

```
map(database="state")
```



图8-2　美国的平面地图

第二层，或者说好市多店铺层，则要通过`symbols()`函数来绘制。它也就是我们在第6章中绘制气泡图所用的函数，使用方法也完全一样，只不过这次传递的是经纬度，而非之前的*x*轴和*y*轴坐标。此外设置`add`为`TRUE`，表示我们是希望在地图上添加其他标记符号，而非创建一个新图表。

```
symbols(costcos$Longitude, costcos$Latitude,
  circles=rep(1, length(costcos$Longitude)), inches=0.05, add=TRUE)
```

图8-3显示了代码的结果。所有的圆圈大小都相同，因为我们将`circles`设置为一个数组，它的长度等于地点的总数，同时我们将`inches`设为了0.05，它规定了各个圆圈的尺寸。如果你希望这些标记再小一点，只需要减小这个值即可。

和之前一样，我们可以调整地图和圆圈的颜色，以便各个地点的显示能更突出，同时让边界线隐入到背景中去，如图8-4所示。现在让我们把这些圆点改为好市多的标准红色，同时把州界线改为浅灰色。

```
map(database="state", col="#cccccc")
symbols(costcos$Longitude, costcos$Latitude, bg="#e2373f", fg="#ffffff",
  lwd=0.5, circles=rep(1, length(costcos$Longitude)),
  inches=0.05, add=TRUE)
```

在图8-3中，未填色的圆圈和地图都是同一种颜色，线条的粗细也一样，所以全都混杂在了一起。但如果正确运用了颜色，我们就能让数据占据视线的焦点。



图8-3 好市多全美店址地图

只用几行代码就能达到这种效果，看起来不坏。很明显，好市多比较倾向于在沿海地带开店，圆点都聚集在加利福尼亚的南北部、华盛顿的西北部以及美国东北部。

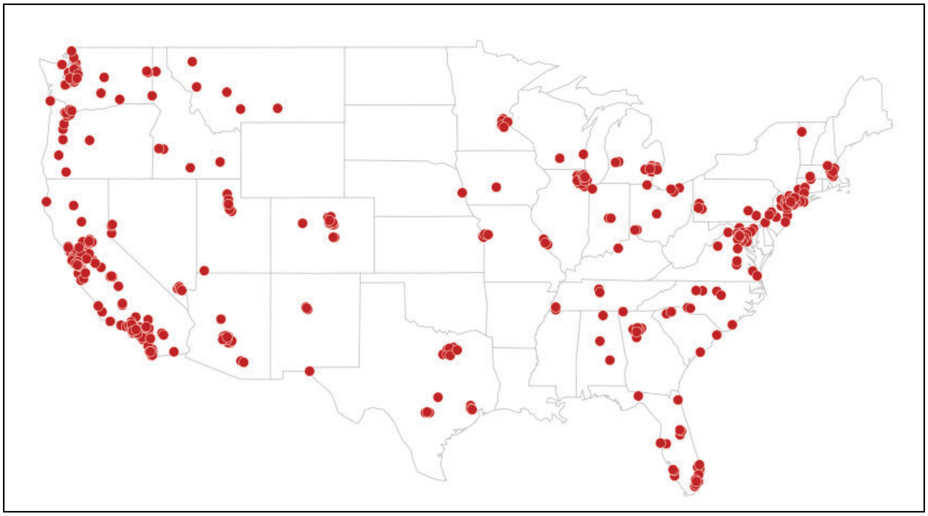


图8-4 为地图上的地点着色

不过，这张图还是有一个遗漏，确切地说应该是两个：阿拉斯加和夏威夷在哪里？它们也是美国的一部分，但在图中没法找到，即使`map()`里已经调用了`"state"`数据库。实际上，这两个州是在`"world"`数据库里面，所以如果想看好市多在阿拉斯加和夏威夷开的店，就需要绘制整个世界地图，如图8-5所示。

```
map(database="world", col="#cccccc")
symbols(costcos$Longitude, costcos$Latitude, bg="#e2373f", fg="#ffffff",
   lwd=0.3, circles=rep(1, length(costcos$Longitude)),
   inches=0.03, add=TRUE)
```



图8-5 好市多店址的世界地图

我知道这有点浪费空间。有很多方法可以解决这个问题，在技术文档里面都可以找到，但最简单的办法就是在Illustrator里面把其他国家的地图删掉，然后把美国放大。

提示 在使用R时，如果遇到了问题，随时可以查阅技术文档。方法是在不熟悉的函数或工具包前面加上一个问号。

现在换一个角度，假设我们只希望显示少数几个州的好市多店址，那么可以用`region`参数来实现这一点。

```
map(database="state", region=c("California", "Nevada", "Oregon",
  "Washington"), col="#cccccc")
symbols(costcos$Longitude, costcos$Latitude, bg="#e2373f", fg="#ffffff",
  lwd=0.5, circles=rep(1, length(costcos$Longitude)), inches=0.05,
  add=TRUE)
```

效果如图8-6所示，我们创建了包括加利福尼亚、内华达、俄勒冈和墨西哥州的地图作为底层，然后在它上面再创建数据层。有一些点并不在这些州里，但它们处于绘图区域中，所以依然会显示。同样地，我们可以在自己喜欢的矢量绘图软件中删除这些点。

2.带有线的地图

有时候，如果地图上点的顺序存在关联，那么可能需要将点连接起来。随着Foursquare等在线定位服务的流行，位置追踪已经并不少见。比较简单的画线方法自然是利用`lines()`函数。为

了方便演示，我们来看一下我作为乌有国政府的特工人员，在7天7夜间的旅行轨迹吧。和往常一样，以载入数据开始，然后先绘制基础的世界地图。

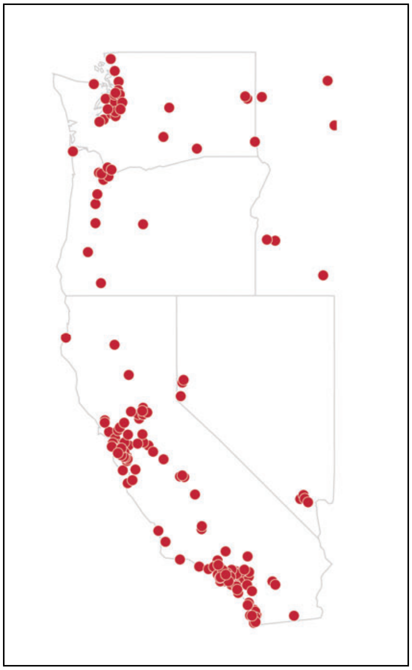


图8-6　选定州中的好市多店址

```
faketrace <-
    read.csv("http://book.flowingdata.com/ch08/points/fake-trace.txt",
      sep="\t")
map(database="world", col="#cccccc")
```

在R的控制台中输入`faketrace`，先看一眼数据帧。你会发现它只显示了两列，分别是纬度和经度，而且只有8个数据点。我们假设这些地点的顺序正是我在这漫长7天中旅行的顺序。

```
   latitude   longitude
1  46.31658    3.515625
2  61.27023   69.609375
3  34.30714  105.468750
4 -26.11599  122.695313
5 -30.14513   22.851563
6 -35.17381  -63.632813
7  21.28937  -99.492188
8  36.17336 -115.180664
```

将这两列插入`lines()`函数，就可以绘制出线条了。同时指定线条的颜色(`col`)和宽度(`lwd`)。

```
lines(faketrace$longitude, faketrace$latitude, col="#bb4cd4", lwd=2)
```

现在再加入圆点。过程和我们之前在好市多一例中所做的一样，效果如图8-7所示。

```
symbols(faketrace$longitude, faketrace$latitude, lwd=1, bg="#bb4cd4",
fg="#ffffff", circles=rep(1, length(faketrace$longitude)), inches=0.05,
add=TRUE)
```

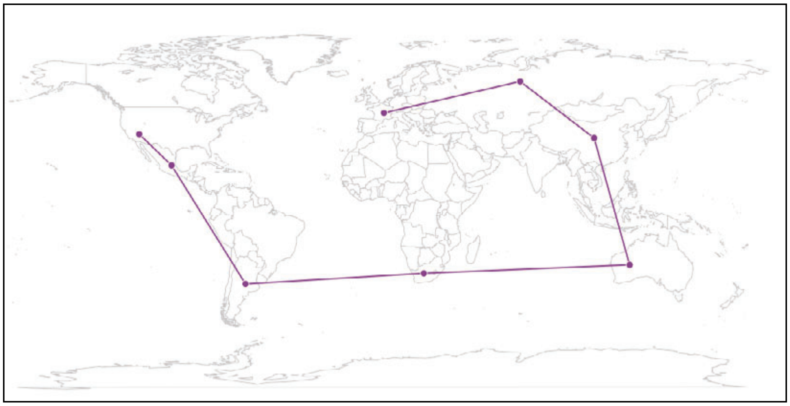


图8-7　绘制位置追踪轨迹

作为特工为乌有国政府奔波了7天7夜之后，我决定洗手不干了。这工作并不像007电影里面表现的那么迷人。不过，我还是为那些我拜访过的国家创建了连接线。从我居住的地方到所有其他国家之间画上线，可能会很有趣，效果如图8-8所示。

```
map(database="world", col="#cccccc")
for (i in 2:length(faketrace$longitude)-1) {
	lngs <- c(faketrace$longitude[8], faketrace$longitude[i])
	lats <- c(faketrace$latitude[8], faketrace$latitude[i])
	lines(lngs, lats, col="#bb4cd4", lwd=2)
}
```



图8-8　绘制与世界各地的连接线

在创建好底层地图之后，我们对数据帧的每个点循环处理，让它们和数据帧最后的那个点之间用线条连接起来。这张图可能信息含量并不高，但也许你会找到合适的机会用上它。举这个例子的目的是希望大家了解，在绘制好地图之后，只要有经纬度坐标，就可以利用R的各种图形函数来随心所欲地绘制你想要的任何东西。

哦，对了，我并没有真的当过乌有国的特工。我只是在开玩笑而已。

8.2.3 有大有小的点

让我们回到真实的数据上来，进入一个比特工恶作剧更有意思的话题。有些时候，我们手上并不只有位置数据，可能还会有其他的数值，例如销售数据或者城市人口等。我们依然是在地图上绘制圆点，但这次可以用上气泡图的原则。

我应该不必再次解释气泡的尺寸是根据面积而非半径来的，对吧？很好。

带有气泡的地图

在本例中，让我们看一下《2008年联合国人类发展报告》中的未成年人生育率，也就是每1000名15 ~19岁年龄段女性中的生育数量。其中的地理坐标由GeoCommons提供。我们希望根据不同国家间生育率的比例来指定各个气泡的大小。

所用到的代码和之前绘制好市多店址地图时所用的差不多，但要记住当时我们只向`symbols()`函数传递了一个向量来控制圆圈的尺寸。而这次，我们会用到生育率的`sqrt()`函数来指定圆点的大小。

```
fertility <-
    read.csv("http://book.flowingdata.com/ch08/points/adol-fertility.csv")
map('world', fill = FALSE, col = "#cccccc")
symbols(fertility$longitude, fertility$latitude,
    circles=sqrt(fertility$ad_fert_rate), add=TRUE,
    inches=0.15, bg="#93ceef", fg="#ffffff")
```

图8-9显示了输出的结果。我们很快就能发现非洲国家的未成年人生育率比较高，而欧洲国家相对较低。单独只看图表，并不清楚每个圆代表多少数值，因为尚未给出图例。在R中用`summary()`可以得到一份概览。

```
summary(fertility$ad_fert_rate)

       Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
       3.20   16.20   39.00   52.89   78.20  201.40    1.00
```

这个数据对我们来说已经不错了，但我们只是受众之一。如果希望其他没有看过此数据的人也能理解图表，就需要更多的解释。我们可以为表现突出（生育率最高和最低）的国家添加注释，可以专门标识出读者最多的国家（在本例中也就是美国），同时还可以提供一段引文，帮助读者理解如何阅读图表。图8-10显示了这些改变。

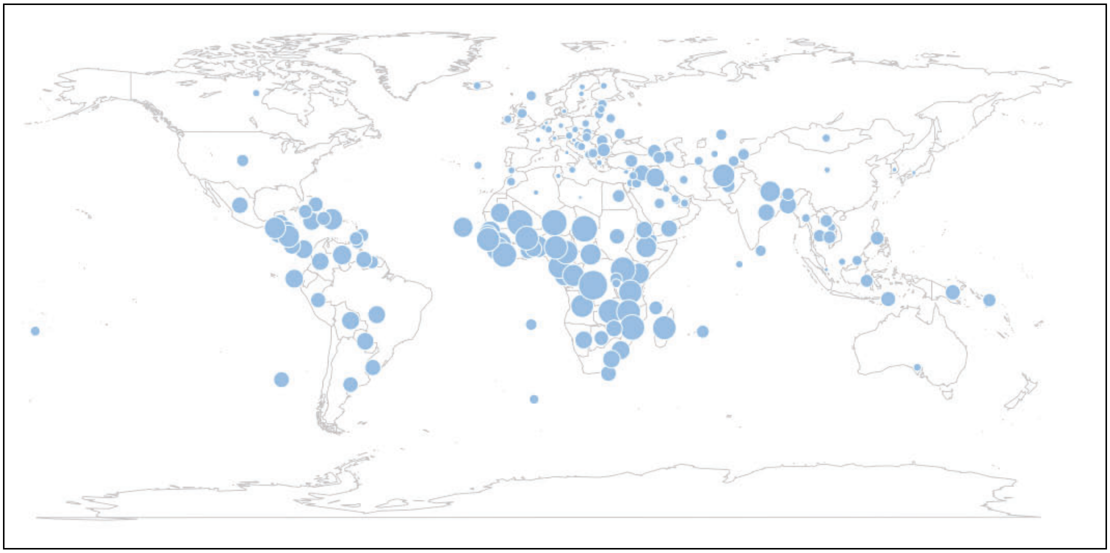


图8-9　全球未成年人生育率

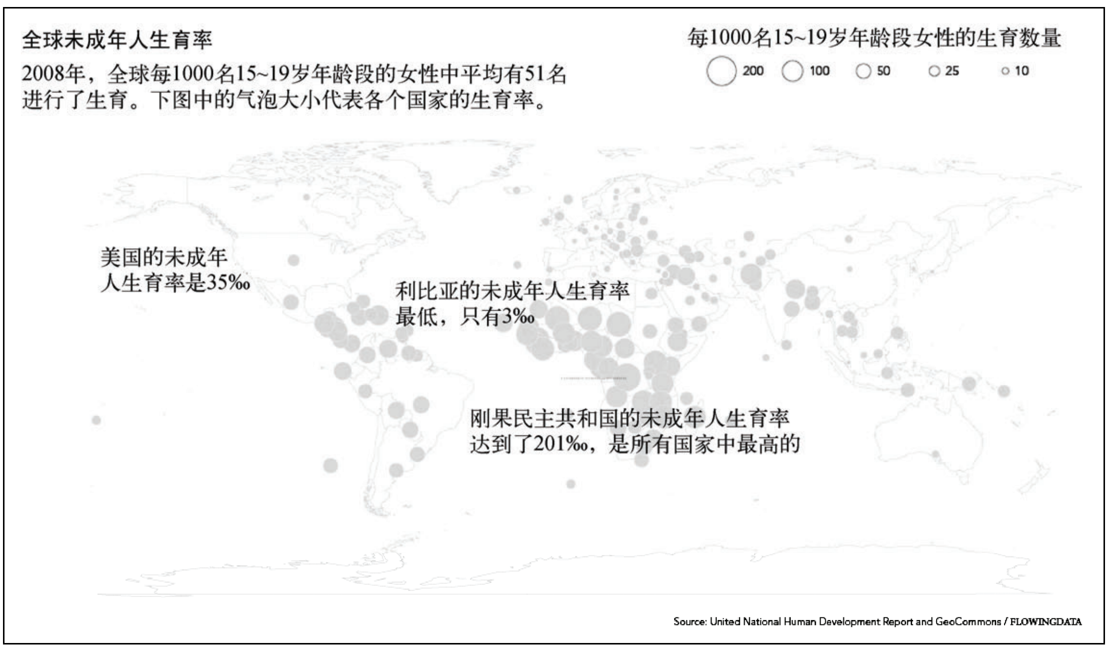


图8-10　为更广泛的读者群对生育率提供清晰的解释

8.3　地区

在地图上绘制圆点为我们带来的价值基本上就是以上这些，因为它们只能代表单个的位置。县、州、国和洲都是带有边界线的完整区域，而且地理数据往往都是通过这种方式生成的。比如说，与单个病人或医院的数据相比，找到某个州或某个国家的居民卫生数据通常都会容易得多。很多时候这些数据都是私人生成的，但聚集起来后，数据就很容易被公开。不管怎么说，我们经常会按这种方式来使用自己手中的空间数据，所以现在看看怎样对它们进行可视化。

根据数据来着色

等值区域图（choropleth map）是最为常见的绘制区域性数据地图的方式。根据某种度量方法，各个地区会根据我们定义的颜色标尺进行着色，如图8-11所示。地区和位置都已经定义好了，所以我们的工作就是选择使用一种合适的颜色标尺。



图8-11　等值区域图的基本框架

我们在上一章曾接触过Cynthia Brewer开发的ColorBrewer，它非常方便于选择颜色，对调色板的设计也有很大帮助。如果我们的数据是延续性的，可能就会希望有类似延续性的颜色标尺，在一个色度（或者多个相似的色度）内由浅到深进行渐变，如图8-12所示。

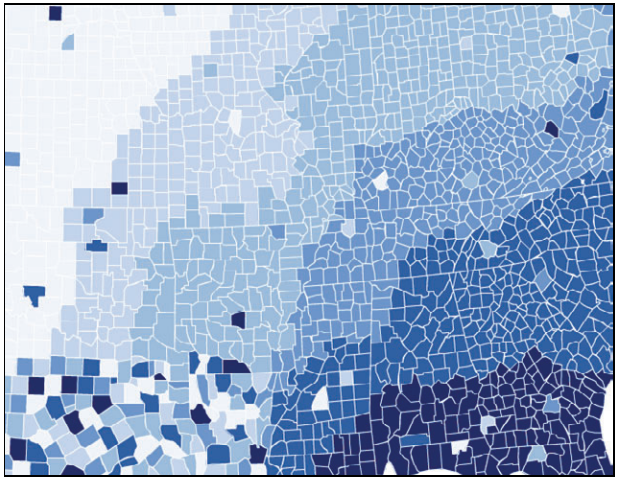


图8-12　ColorBrewer实现的连续性配色

如果数据代表的是正反两种品质，例如好和坏、高于或低于临界值，那么相互背离的配色方案可能会更为合适，如图8-13所示。

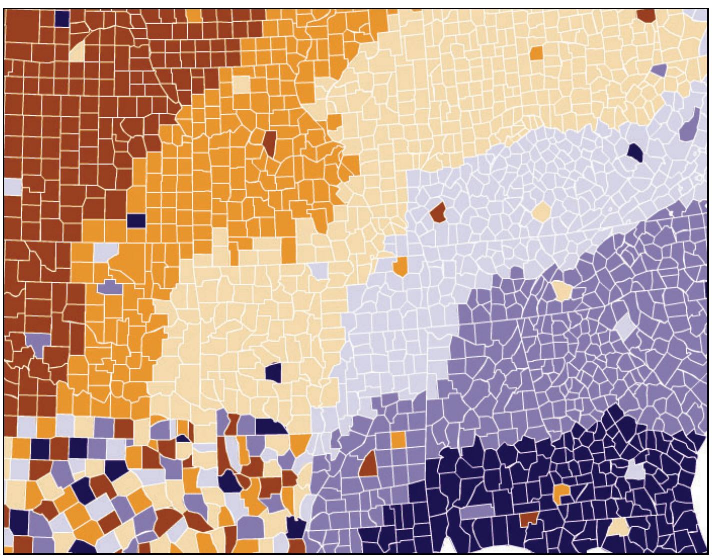


图8-13　ColorBrewer实现的相互背离的配色

最后，如果数据分属于不同的类，那么就需要为每一类填充唯一的颜色（见图8-14）。

当我们有了自己的配色方案之后，还有两件事情要做。第一件是决定如何让选择的颜色与数据的范围相适配，第二件是根据我们的选择为每个地区安排颜色。在下面的例子中，我们可以用Python和可缩放矢量图形（Scalable Vector Graphics，SVG）来完成以上两件事情。

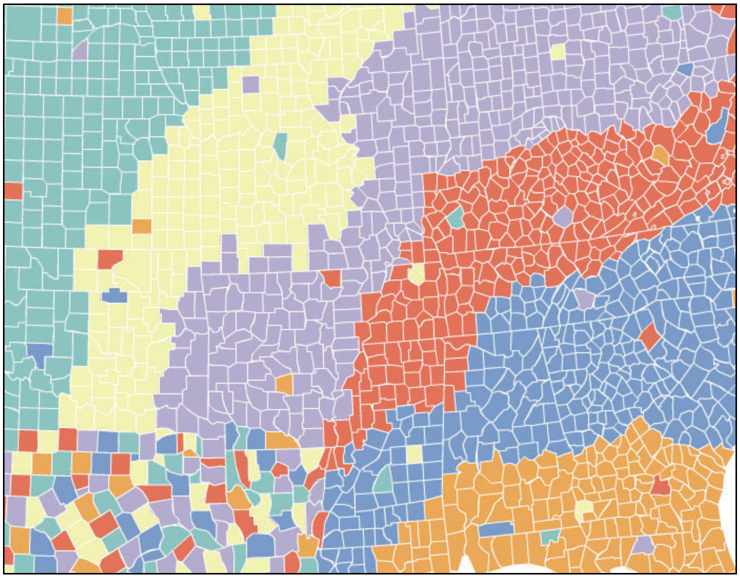


图8-14　ColorBrewer实现的定性配色（另见彩插图8-14）

1. 绘制县的地图

美国劳工统计局每个月都会提供县级别的失业率数据。我们可以从其官方网站上下载从数年前到最近的数据。不过，劳工统计局提供的数据表现形式有点过时了，操作也很麻烦，所以为了简单起见（也为了避免其网站改版），大家可以在http://book.flowingdata.com/ch08/regions/unemployment-aug2010.txt下载数据。数据共有6列，第一列是劳工统计局专用的编码，其后两列是唯一的ID，用于标识各个县。第4列和第5列分别是各县的名称以及月份，最后一列则是各郡县失业人数的估算百分比。在本例中，我们只用关心县的ID（也就是联邦资料处理标准码，FIPS codes）和失业率即可。

现在来画地图。在上一章中，我们是用R生成的底层地图，这次我们可以试试用Python和SVG来做到这一点。前者用于处理数据，而后者用于绘制地图。不过，我们也不需要完全从头开始。从维基共享资源（Wikimedia Commons）可以获得一张空白的地图，地址是http://commons.wikimedia.org/wiki/File:USA_Counties_with_FIPS_and_names.svg，如图8-15所示。该页面链向4种尺寸的PNG格式地图和1个SVG格式地图，我们要的是SVG格式。下载SVG文件并存储为counties.svg，目录与你保存失业率数据的文件夹相同。

如果你不熟悉SVG，只需要知道它实际上就是一种XML文件。它是带有标签的文本，我们可以在文本编辑器里对其进行编辑，就像编辑HTML文件那样。网页浏览器或者图片浏览器会读取XML，而XML告诉浏览器显示什么内容，例如绘制什么形状、使用什么颜色。

我们可以验证这一点。用文本编辑器打开这个地图SVG文件，看看你将要面对的是什么。其中大部分都是SVG的声明以及一些套话，现在完全不需要关心。

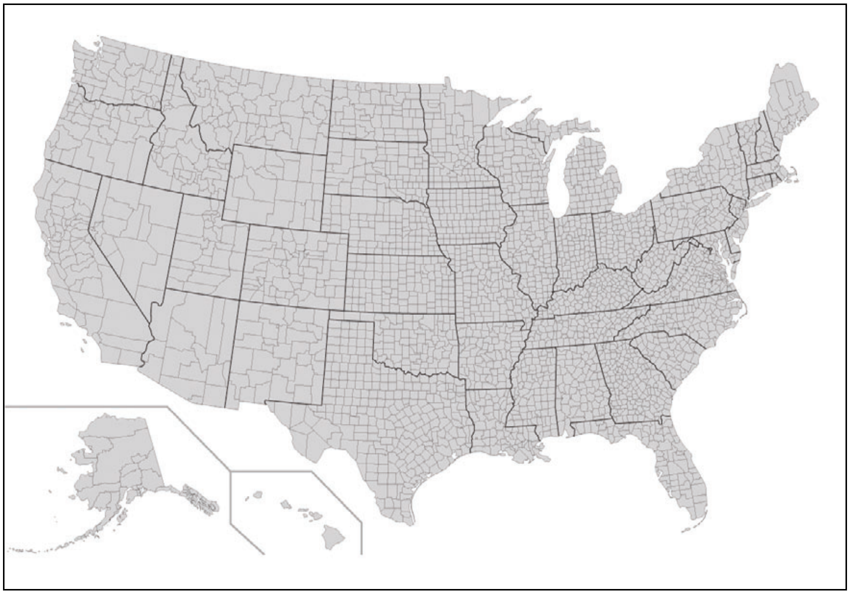


图8-15　来自维基共享资源的空白美国县地图

往下滚动页面，可以看到一些`<path>`标签，如图8-16所示。在一个标签内的所有数字就指定了县的边界。我们现在不要动它们。我们感兴趣的是修改每个县的填充色，以便适配其对应的失业率。要做到这一点，必须修改这些路径的样式。

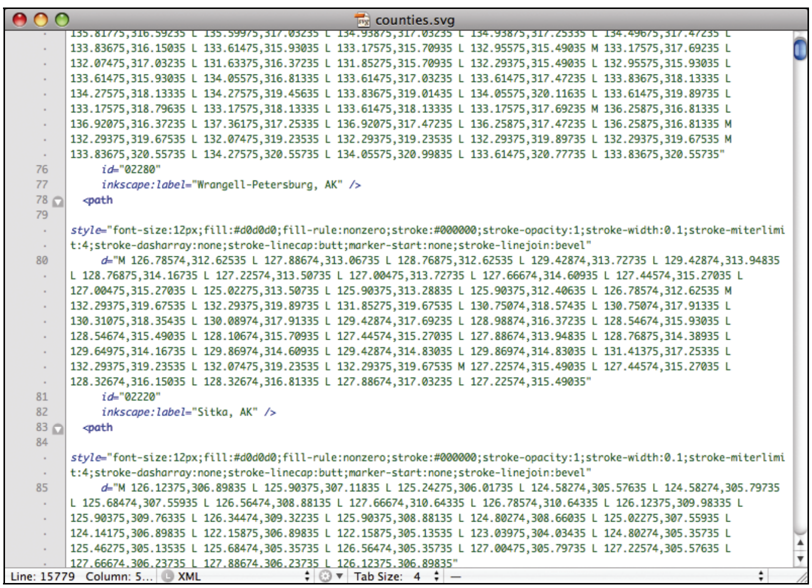


图8-16　SVG文件中指定的路径

提示　SVG文件就是XML文件，很容易在文本编辑器里面修改。这也意味着我们可以解析SVG代码，通过程序来修改。

注意到每个`<path>`都是以`style`开始的了吗？那些写过CSS的人可能立刻就认出来了。其中有一个`fill`参数，其后跟着十六进制的颜色值，所以如果我们在SVG文件里修改它们，就能在输出的图片中调整颜色了。你可以手工去一个个地修改，但全美国有3000多个县。这要花的时间可就长了。现在让我们找找老朋友Python工具包Beautiful Soup，它能让XML和HTML的解析变得相对容易一些。

在存储SVG地图和失业率数据的文件夹里新建一个空白文件，并改名为colorize_svg.py。我们需要导入CSV数据文件，然后用Beautiful Soup进行解析，所以第一步先载入需要的工具包。

```
import csv
from BeautifulSoup import BeautifulSoup
```

然后打开CSV文件并存储之，以便我们用`csv.reader()`循环处理每一行。请注意，在`open()`函数里面的`"r"`只是表示我们希望打开文件并读取（read）里面的内容，而不是往里面写入更多新的行。

```
reader = csv.reader(open('unemployment-aug2010.txt', 'r'), delimiter=",")
```

现在再载入那个空白的SVG县地图。

```
svg = open('counties.svg', 'r').read()
```

太好了，我们已经载入了创建等值区域图所需要的所有东西。现在的挑战在于你得把数据和SVG连接起来。这两者之间的共性是什么？给你一个提示：它肯定和每个县的唯一ID有关，我在之前已经提过。如果你猜的是FIPS码，那么恭喜你答对了。

SVG文件中的每一条路径都有唯一的ID，正好是其所属州及所属县的FIPS码的混合。而在失业率数据的每一行里也有州和县的FIPS码，不过是分开的。比如说，阿拉巴马州奥拓加（Autauga）县的州FIPS码是01，县FIPS码是001。而SVG中该路径的ID则是二者的结合：01001。

我们需要存储失业率数据，以便通过FIPS码来检索每一个县的失业率，就和循环处理每一条路径一样。如果大家有点糊涂了，先跟着我的思路，到后面看到实际代码就会更加清楚了。这里主要是说FIPS码就是我们SVG和CSV之间的纽带，对这一点可以加以利用。

提示　SVG文件中的路径（尤其是地理路径）通常都会有唯一的ID。这种ID不一定是FIPS码，但规则是一样的。

要想存储失业率数据，便于后面通过FIPS码读取，我们要用到Python中的“字典”这一概念。它能让我们通过关键词来存储和检索数值。在本例中，我们的关键词就是相结合后的州和县的

FIPS码，如以下代码所示。

```
unemployment = {}
min_value = 100; max_value = 0
for row in reader:
try:
    full_fips = row[1] + row[2]
    rate = float( row[8].strip() )
    unemployment[full_fips] = rate
except:
    pass
```

下面用Beautiful Soup来解析SVG文件。绝大多数标签都有一个开始标签和一个结束标签，但也有一些自关闭标签，这些需要专门指定。之后用`findAll()`函数来检索地图中的所有路径。

```
soup = BeautifulSoup(svg, selfClosingTags=['defs','sodipodi:namedview'])
paths = soup.findAll('path')
```

然后把颜色（我从ColorBrewer得到的）存储进一个Python列表。这是一个延续性的配色方案，带有多个色度，由紫色渐到红色。

```
colors = ["#F1EEF6", "#D4B9DA", "#C994C7", "#DF65B0", "#DD1C77", "#980043"]
```

我们开始接近高潮了。正如我刚才所说，我们要修改SVG文件里每一条路径的`style`属性。现在我们只对填充色感兴趣，但为了让事情更容易点，我们可以替换整个`style`而不是仅仅只替换掉填充色。我将位于`"stroke"`后面的十六进制数值改成了`#ffffff`（也就是白色）。这样一来边界线就由当前的灰色变成了白色。

```
path_style = 'font-size:12px;fill-rule:nonzero;stroke:#fffff;stroke-
opacity:1;stroke-width:0.1;stroke-miterlimit:4;stroke-
dasharray:none;stroke-linecap:butt;marker-start:none;stroke-
linejoin:bevel;fill:'
```

与此同时，我还把`fill`挪到了代码末端，并且把它的值留为空。因为这一部分需要根据各个县的失业率来定。

终于，我们要开始修改颜色了。我们可以循环处理每一条路径（州边界线和夏威夷、阿拉斯加的分隔线之外）及其对应的颜色。如果失业率高于10%，就用较深的颜色；如果低于2，则用最浅的颜色。

```
for p in paths:

    if p['id'] not in ["State_Lines", "separator"]:
        # pass
try:
    rate = unemployment[p['id']]
except:
    continue
```

```
if rate > 10:
    color_class = 5
elif rate > 8:
    color_class = 4
elif rate > 6:
    color_class = 3
elif rate > 4:
    color_class = 2
elif rate > 2:
    color_class = 1
else:
    color_class = 0

color = colors[color_class]
p['style'] = path_style + color
```

最后一步是用`prettify()`来输出SVG文件。该函数可以把我们的`soup`转化为一个浏览器能够理解的字符串。

```
print soup.prettify()
```

▶大家可以在这里获得完整的脚本：http://book.flowingdata.com/ch08/regions/colorize_svg.py.txt。

现在还需要做的就是运行Python脚本，并将输出存储为一个新的SVG文件，名字可以叫做colored_map.svg（见图8-17）。

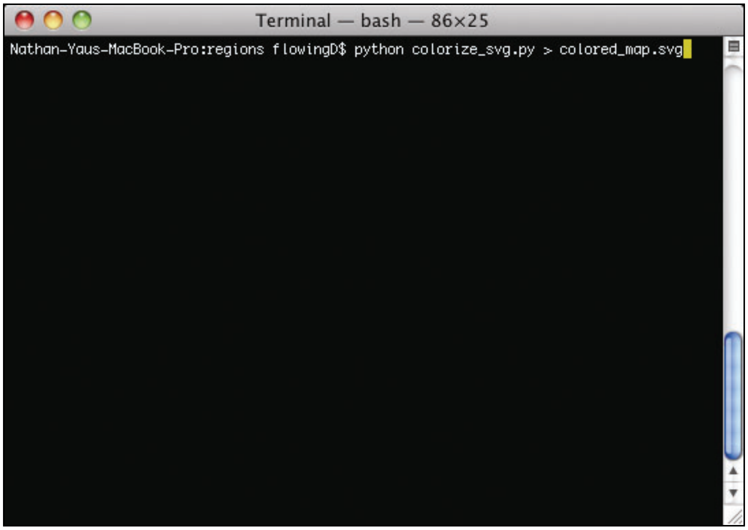


图8-17　运行Python脚本并将输出存储为一个新的SVG文件

8

在Illustrator或者Firefox、Safari、Chrome等现代浏览器中打开我们新鲜热辣的等值区域图，就能看到辛苦劳动的成果了，如图8-18所示。现在很容易看出来在2010年8月美国有哪些地方的失业率较高。很明显，西海岸大部分地区和东南部许多地区的失业率较高，阿拉斯加和密歇根的形势也很严峻。而美国中部大多数县的失业率相对较低。

困难的部分已经过去，我们现在可以尽情对地图进行视觉上的优化了。在Illustrator中打开SVG文件，修改边框颜色和宽度、添加注释，绘制出一幅完整的图表以便更多人能够理解。（提示：图例还是必不可少的。）

最棒的是，这段代码是可重用的，我们可以将其应用到其他使用FIPS码的数据集中去。或者对于同样的数据集，我们还可以自由改变配色方案，从而设计出适合自己数据风格的地图。

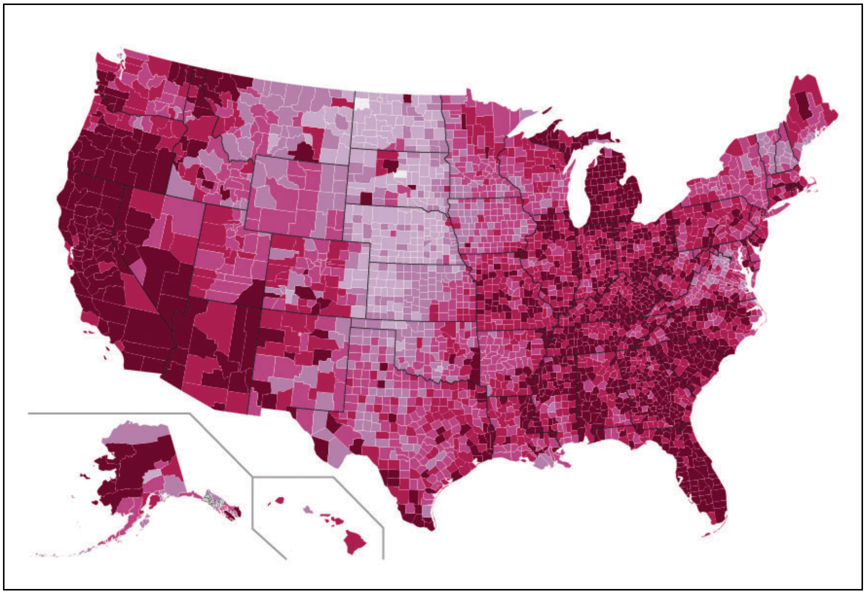


图8-18　显示失业率的等值区域图

根据手中的数据，我们还可以修改临界值来调整每个地区的颜色。之前的例子中，我们使用了相等的临界值，各个地区按6种色值进行着色，每2个百分比一个等级。失业率超过10%的所有县都归为同一个等级，然后失业率在8%和10%之间的是一个等级，之后是6%~8%，依此类推。另一种定义临界值的常用方法就是利用四分位数，也就是只有四种颜色，每一种颜色代表地区总数的1/4。

比如说，对于这些失业率的下四分点、中四分点和上四分点分别是6.9%、8.7%和10.8%。这表示有1/4的县失业率低于6.9%、1/4的县失业率在6.9%和8.7%之间、1/4的县失业率在8.7%和10.8%之间，最后1/4的县失业率高于10.8%。要想做到这一点，可以按以下代码修改颜色列表。这是一个紫色的配色方案，每1/4对应一种颜色。

```
colors = ["#f2f0f7", "#cbc9e2", "#9e9ac8", "#6a51a3"]
```

然后用刚才的几个四分点在for循环中修改着色条件。

```
if rate > 10.8:
    color_class = 3
elif rate > 8.7:
    color_class = 2
elif rate > 6.9:
    color_class = 1
else:
    color_class = 0
```

像之前那样运行脚本并存储，得到图8-19。请注意现在着色的县数量稍有增多。

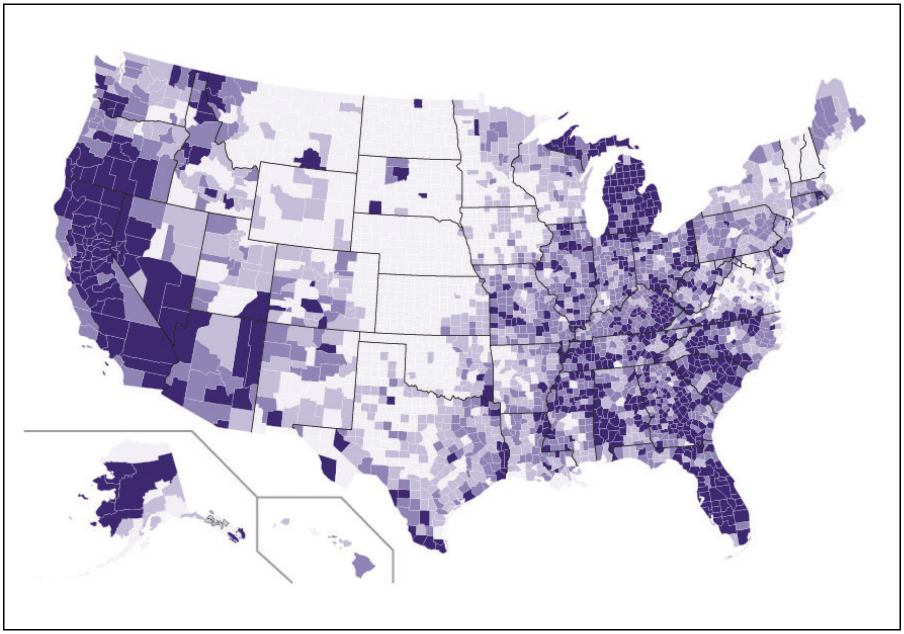


图8-19　按四分位数区分的失业率

为了提升代码的可用性，我们可以通过程序来计算四分位数，而不是依靠手写代码。这一过程在Python里面非常简单。把数值存储为一个列表，按由小到大排序，然后找到第1/4、1/2和3/4的几个数值。具体来说，在本例中我们可以修改colorize_svg.py中的第一个循环，使之只存储失业率。

```
unemployment = {}
rates_only = [] # To calculate quartiles
min_value = 100; max_value = 0; past_header = False
for row in reader:
    if not past_header:
```

8

```
        past_header = True
        continue

    try:
        full_fips = row[1] + row[2]
        rate = float( row[5].strip() )
        unemployment[full_fips] = rate
        rates_only.append(rate)
    except:
        pass
```

然后我们可以对数组进行排序，并找出那几个四分位数。

```
# 四分位数
rates_only.sort()
q1_index = int( 0.25 * len(rates_only) )
q1 = rates_only[q1_index]    # 6.9

q2_index = int( 0.5 * len(rates_only) )
q2 = rates_only[q2_index]    # 8.7

q3_index = int( 0.75 * len(rates_only) )
q3 = rates_only[q3_index]    # 10.8
```

现在我们不必手动在代码里输入6.9、8.7和10.8这几个数值了，而可以用q1、q2和q3来代替它们。通过程序来计算这些值的好处在于，我们可以方便地将代码重用于不同的数据集，只需变动CSV文件即可。

选择何种色彩标尺取决于我们手中的数据，以及希望传达何种讯息。对本例中的数据集，我更喜欢线性标尺，因为它能更好地表现出分布，并且突出显示出整个国家中失业率相对较高的地区。以图8-18为基础，我们可以添加图例、标题和引文，从而得到一幅更加完善的图表，如图8-20所示。

2. 绘制国家的地图

上例中为县着色的过程并不只针对于县这一级别，我们还可以重复同样的步骤为州或者国家着色。你需要的只是每个地区带有唯一ID的SVG文件（从维基百科上很容易获取），以及与ID适配的数据即可。现在让我们用来自世界银行的公开数据来试一试。

提示　世界银行是按国家划分的最完整的人口统计数据来源之一。它一般都是我的首选。

让我们看看2008年各国获得安全饮用水源的城市居民百分比数据。大家可以从世界银行数据网站下载到Excel文件：http://data.worldbank.org/indicator/SH.H2O.SAFE.UR.ZS/countries。为方便起见，也可以下载经过我精简后的CSV数据文件，地址是http://book.flowingdata.com/ch08/worldmap/water-source1.txt。其中有一些国家的数据丢失了，这在国家级别的数据中很常见。我

已经在CSV文件中删除了这些行。

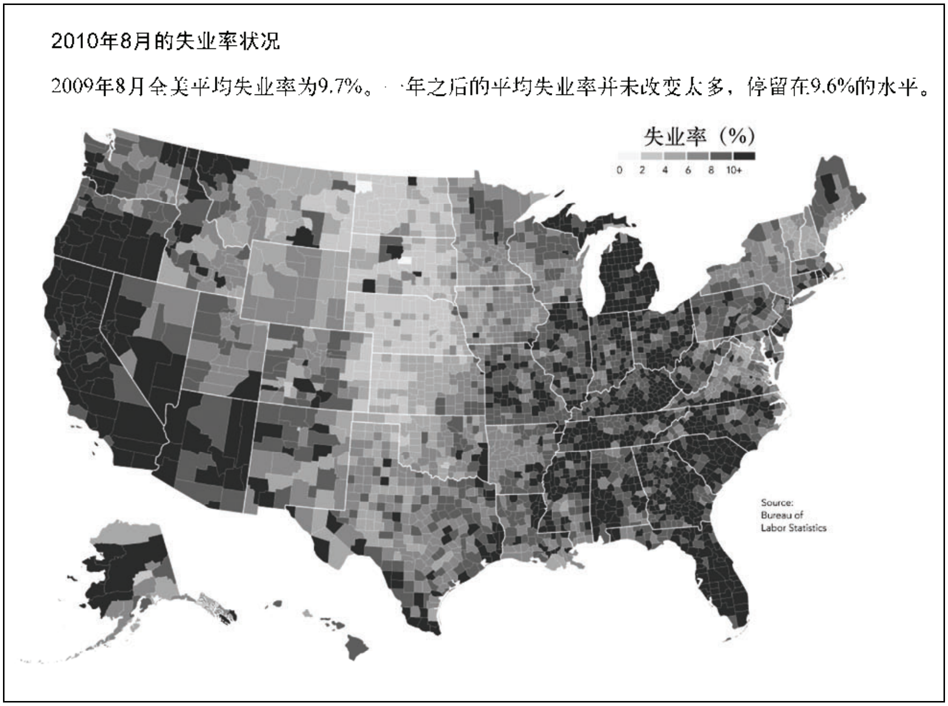


图8-20 带有标题、引文和图例的最终地图

数据共有7列。第一列是国家名称，第二列是国家代码（这是否能变成我们的唯一ID？），而后五列则是1990—2008年的百分比。

要找底层地图，可以再次求助维基百科。在搜索SVG世界地图时，我们可以找到很多版本，统一用这一张：http://en.wikipedia.org/wiki/File:BlankMap-World6.svg。下载完整分辨率（Full resolution）的SVG文件，存储到与存放数据相同的文件夹。如图8-21所示，这是一个空白的世界地图，颜色是灰色，带有白色的边界线。

在文本编辑器里面打开SVG文件。自然，它是一个XML格式的文本，但是内容的格式与我们上个例子稍有不同。路径没有ID，而且`style`属性也没有用上。不过，`path`标签里有一个类名，看上去好像是国家代码，但只有两个字母。而世界银行数据中的国家代码有三个字母。

世界银行的资料显示，他们使用的是ISO 3166-1三字母码。而来自维基百科的SVG文件用的则是ISO 3166-1二字母码。我知道这些名词看起来很吓人，但不用慌张，我们根本不需要记住这些。你只需要知道维基百科为此提供了一份转换表格，地址是http://en.wikipedia.org/wiki/ISO_3166-1。我将这份表格复制粘贴进了Excel，并将重要内容存成了一个文本文件，其中一列是二字母码，另一列是三字母码。下载地址是http://book.flowingdata.com/ch08/worldmap/country-codes.txt。我们将用这份表格在两种国家代码间切换。

8

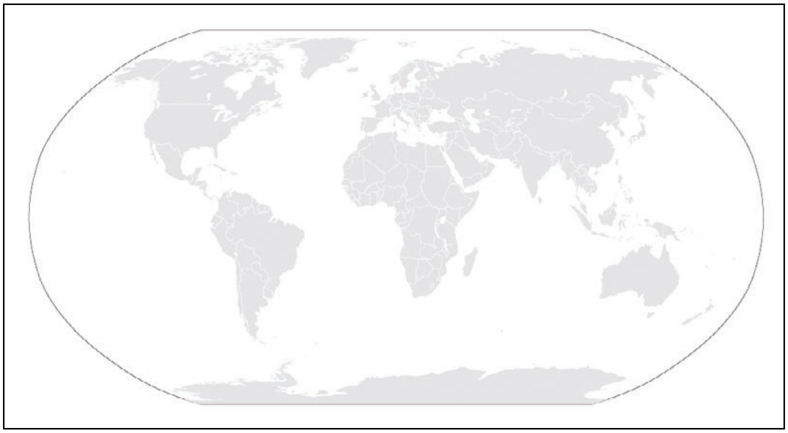


图8-21　空白的世界地图

至于修改各个国家的样式，我们也可以稍微换一种做法。这次不再直接修改path标签里的属性了，试试用路径外部的层叠样式表（CSS，Cascading Style Sheets）来为各个地区着色。

在SVG和CSV文件相同目录下创建一个名为generate_css.py的文件。再次导入CSV工具包来载入SCV文件中的数据，包括国家代码和获得水资源的人口百分比。

```
import csv
codereader = csv.reader(open('country-codes.txt', 'r'), delimiter="\t")
waterreader = csv.reader(open('water-source1.txt', 'r'), delimiter="\t")
```

然后存储国家代码，以便将三字母码转变为二字母码。

```
alpha3to2 = {}
i = 0
next(codereader)
for row in codereader:

    alpha3to2[row[1]] = row[0]
```

这样能把代码存储进一个Python字典，其中三字母码是关键词，而二字母码是具体的值。现在和上个例子一样，循环处理每一行水数据，并根据当前国家的具体值设置一个颜色。

```
i = 0
next(waterreader)
for row in waterreader:

    if row[1] in alpha3to2 and row[6]:
        alpha2 = alpha3to2[row[1]].lower()
        pct = int(row[6])
```

```
        if pct == 100:
            fill = "#08589E"
        elif pct > 90:
            fill = "#08589E"
        elif pct > 80:
            fill = "#4EB3D3"
        elif pct > 70:
            fill = "#7BCCC4"
        elif pct > 60:
            fill = "#A8DDB5"
        elif pct > 50:
            fill = "#CCEBC5"
        else:
            fill = "#EFF3FF"
        print '.' + alpha2 + ' { fill: ' + fill + ' }'

    i += 1
```

这段脚本执行了以下几个步骤：

(1) 避开CSV文件的头部；

(2) 开始循环读取水数据；

(3) 如果有二字母码对应于CSV的三字母码，并且该国家有2008年的数据，那么就找到了对应的二字母码；

(4) 根据百分比数据选择合适的填充色；

(5) 为每行数据都输出一行CSS。

运行generate_css.py并将输出保存为style.css。该CSS的前几行应该类似这样：

```
.af { fill: #7BCCC4 }
.al { fill: #08589E }
.dz { fill: #4EB3D3 }
.ad { fill: #08589E }
.ao { fill: #CCEBC5 }
.ag { fill: #08589E }
.ar { fill: #08589E }
.am { fill: #08589E }
.aw { fill: #08589E }
.au { fill: #08589E }
...
```

这是标准的CSS。例如第一行会把所有类名为af的路径的填充色改为#7BCCC4。

在文本编辑器里面打开style.css，并复制所有内容。然后打开SVG地图，将复制的内容粘贴到oceanxx的花括号下面，大概在第135行。现在我们已经创建了一幅世界范围的等值区域图，并根据各国获得安全饮用水源的人口百分比着了色，效果如图8-22所示。深蓝色表示100%，而浅绿色代表百分比较低。依然呈灰色的国家表示其数据未知。

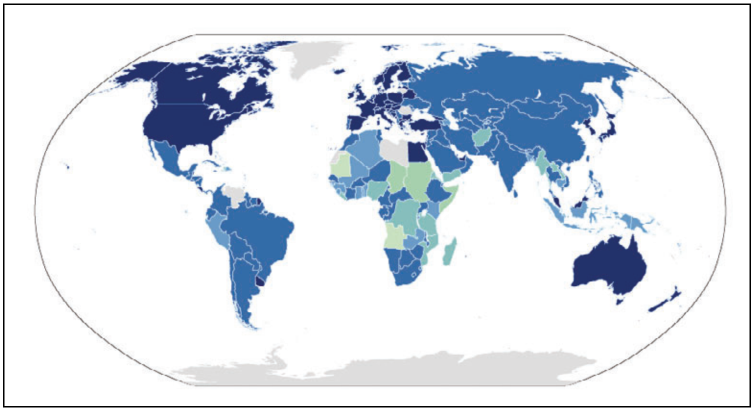


图8-22　显示了安全饮用水资源的世界范围等值区域图

最棒的是，现在你可以下载世界银行的任意数据集（有很多），而只需改动几行代码就能快速为它们创建等值区域图了。要想让图8-22的图形再漂亮点，我们可以继续用Illustrator打开SVG文件进行编辑。这幅地图主要还需要一个标题和用于说明各颜色意义的图例，最终效果如图8-23所示。

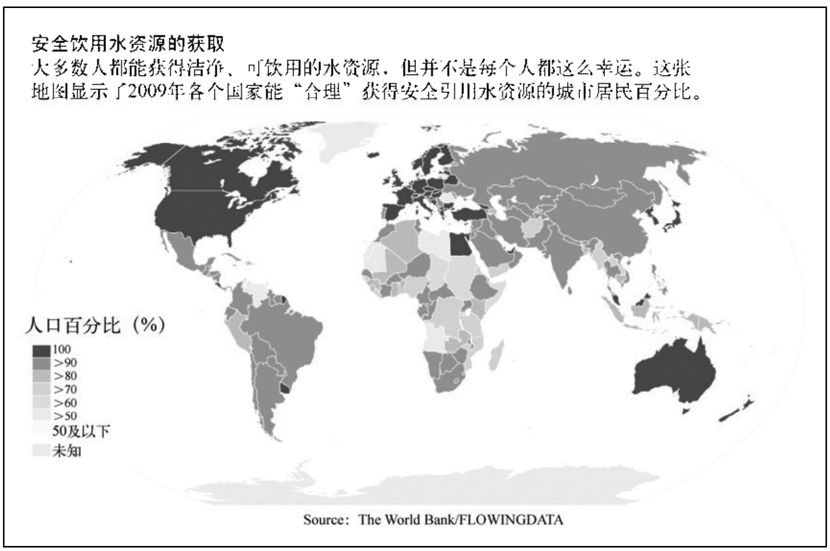


图8-23　完成的世界地图（另见彩插图8-23）

8.4　跨越空间和时间

之前的那些例子让我们能够对多种数据类型（既有定性的也有定量的）进行可视化。我们可以根据要讲的故事选择合适的颜色、类型和标记符号，为地图添加注释突出显示具体的地区或热点，还可以放大到以国家或县为单位。不过，这还不是全部，我们还能更进一步。如果再引入另

一维度的数据，我们就能看到涵盖时间和空间的变化。

在第4章我们用抽象的线条和图表对时间进行了可视化，这很有用。但如果在数据中附加了位置信息，就能通过地图更为直观地了解到模式和变化，而且观察那些物理距离相近的地区分类和群组也容易得多。

最棒的是，在涵盖时间和空间的数据可视化中，我们可以用上已经学到的理论和技巧。

8.4.1　系列组图

我们在第6章中见识过这一技巧。当时可视化的是跨越多个分类的数据之间的关系，但其实它同样也适用于空间数据，如图8-24所示。与之前的小型柱状图表不同，这次用的是小尺寸的地图，每一张地图代表一个时间片段。将这些地图从左往右或从上往下排列，我们的视线自然就会随之移动，了解到发生了哪些变化。

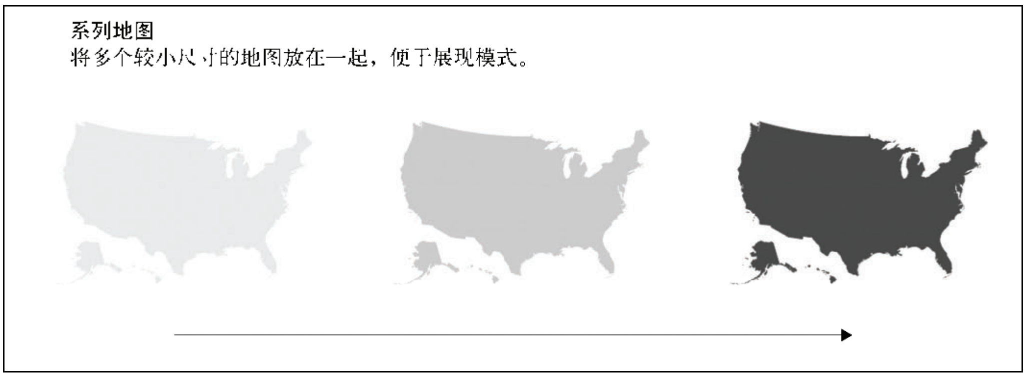


图8-24　地图系列组图

比如说，在2009年底我设计了一幅显示美国失业率的图表（见图8-25）。实际上我用的就是上一节中大家看到的代码（稍有变化），但我将其应用到了多个时间片段上。

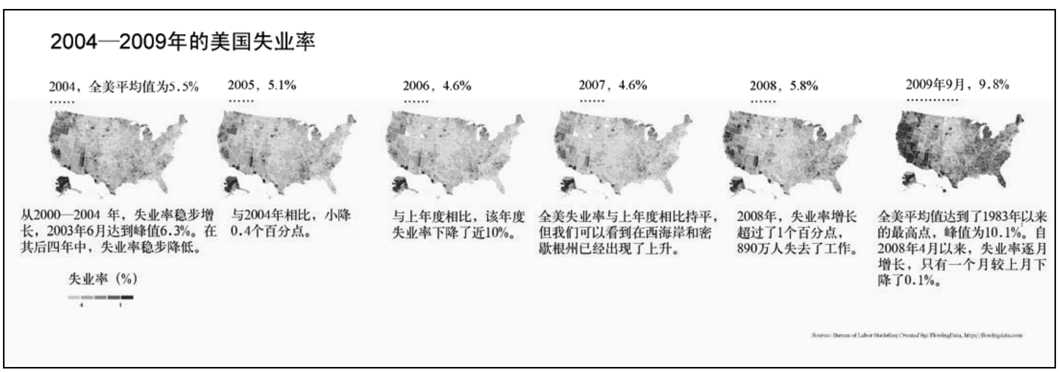


图8-25　2004—2009年的美国失业率

很容易看到2004—2006年的变化，如图8-26所示。全美的平均失业率在这一阶段呈现出下降趋势。

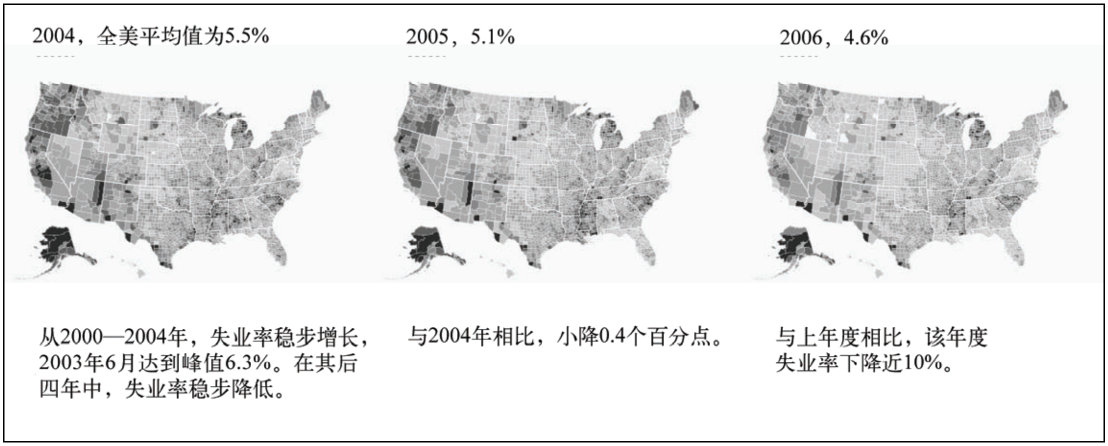


图8-26　2004—2006年的美国失业率

然后到了2008年（见图8-27），我们开始看到失业率有了抬头的趋势，尤其是在加利福尼亚州、俄勒冈州、密歇根州，以及美国东南部的一些县。

而到了2009年，情况大不相同了，如图8-28所示。全美平均值增长了4个百分点，而县的颜色变得非常深。

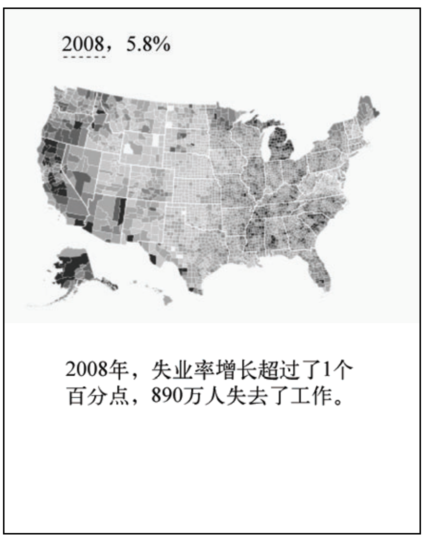


图8-27　2008年的失业率

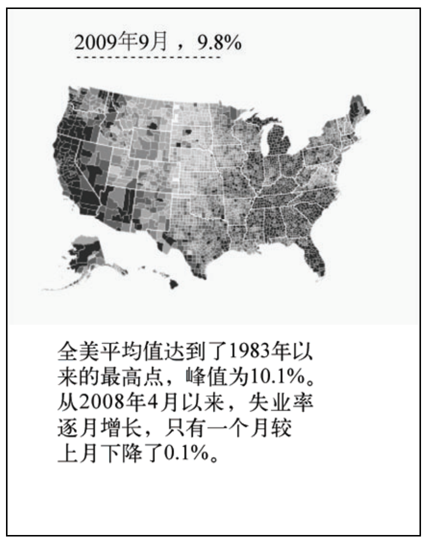


图8-28　2009年9月的失业率

这是我发布在FlowingData上的图表中最受欢迎的图表之一，因为人们能很清楚地看到这种在数年相对停滞后突发的戏剧性转变。我同时也用到了OpenZoom Viewer，它能让读者放大到高分辨率图片，以便着重关注自己所在地区的失业率是如何变化的。

> ▶ 当高分辨率图片过大、无法在一屏中完全显示时，将图片放到OpenZoom Viewer（http://open-zoom.org）里面去会很有帮助，读者既能看到缩小后的全图，又能随时放大观察细节。

我也可以将这些数据可视化为一幅时间序列图表，每一条线代表一个县。但是美国有3000多个县，图表会非常密集，而且如果它不可交互，没人能说出哪条线代表哪个县。

8.4.2　抓住差额

并不是只能创建系列地图才能表现出变化。有时候在一幅地图中只对变化进行可视化反而会更有意义。这种做法能节省空间，而且它突出的是变化而非单个的时间片段，如图8-29所示。



图8-29　关注变化本身

这次我们从世界银行下载城市居民的人口数据，与之前获取安全水资源一例中的数据比较相似。每一行代表一个国家，每一列代表一年。然而，城市人口数据是指一个国家居住在城市地区的人数的粗略计数，基于这种计数创建的等值区域图，必然会突出显示那些较大的国家，因为这些国家的总人口数更多。所以要想用两幅地图来表现2005年和2009年的城市人口差距，就必须把绝对数值改为比例，否则根本没有意义。而要做到这一点，我们必须下载2005年和2009年所有国家的人口数据，然后做一些运算。这个步骤并不太难，但总归需要额外的工作。此外，如果变化并不明显，在多幅地图中也是很难被发现的。

与其这样，我们不如找出其中的差额部分，并将其显示在一张地图上。你可以在Excel里很容易地计算出来，或者修改之前的Python脚本，然后绘制出单独的地图，如图8-30所示。

在对差额进行可视化之后，很容易看出来哪些国家的变化较大，哪些国家的变化较小。与之相比，图8-31显示了2005年各个国家居住在城市的人口比例。

而图8-32显示了2009年的这一数据。它看上去和图8-31很相似，我们很难察觉出其中的不同。

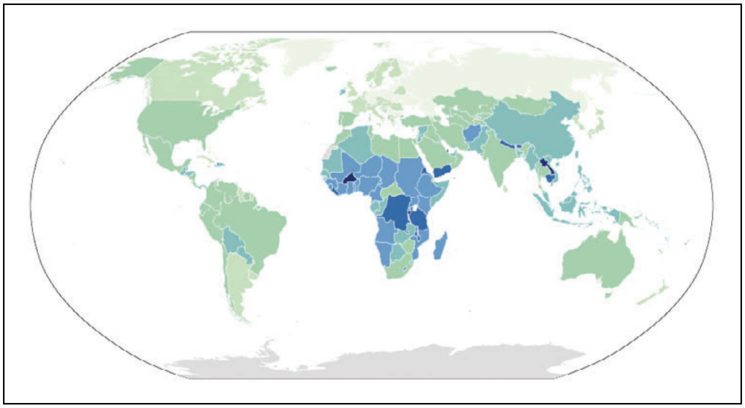


图8-30　2005—2009年间城市人口的变化（另见彩插图8-30）

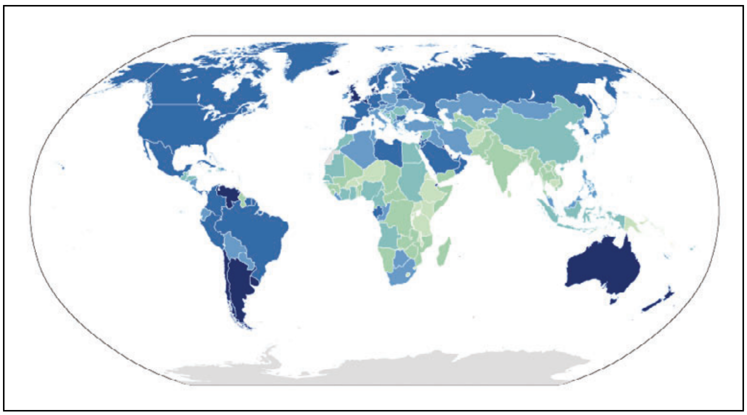


图8-31　2005年居住在城市地区的人口百分比

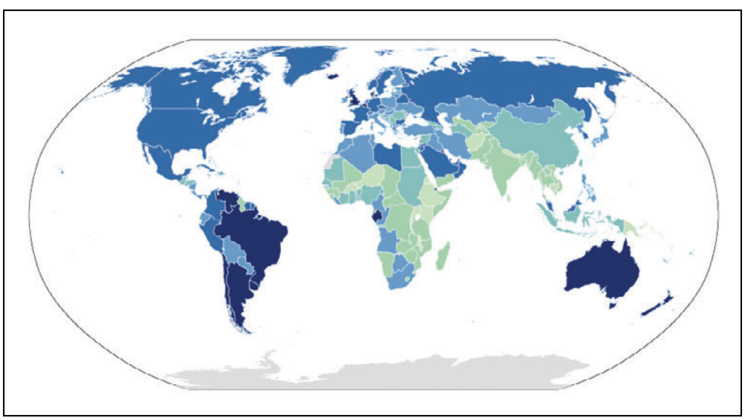


图8-32　2009年居住在城市地区的人口百分比

对于这个例子来说，很明显单幅地图反而更能传达信息。我们必须让读者在理解城市人口的变化时尽量少花脑筋。很容易看出，尽管与世界其他国家相比，非洲许多国家居住在城市的人口比例较低，但最近几年中他们在这方面的增长是最快的。

别忘了再加上图例、数据来源和标题，以面向更广泛的读者群体，如图8-33所示。

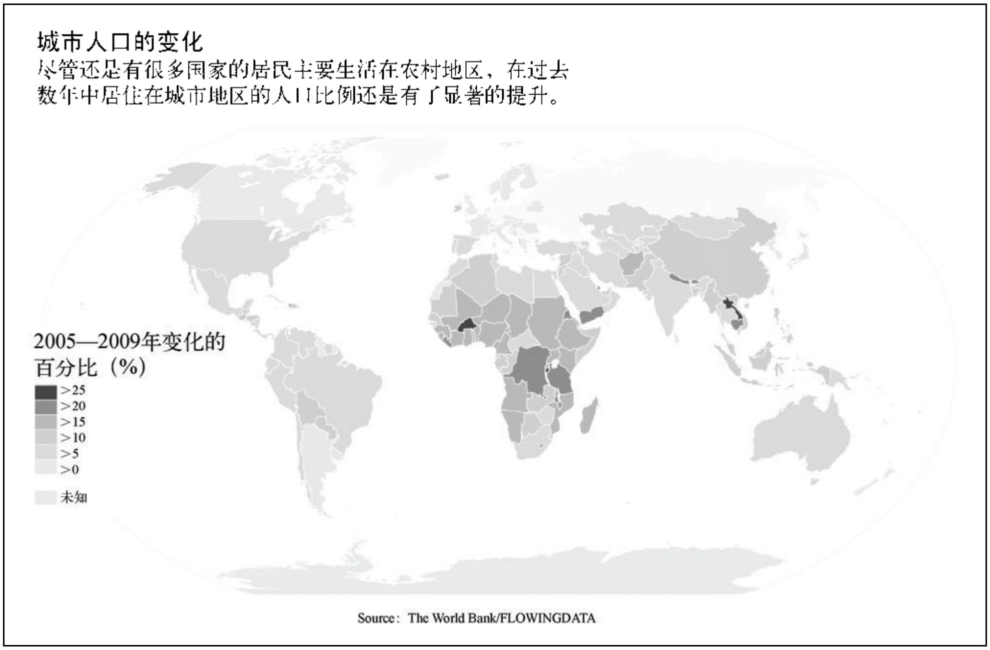


图8-33　带有说明的表现差额的地图

8.4.3　动画

更明显的一种可视化空间和时间变化的方式是让数据“动”起来。不再是通过单个地图显示各个时间片段，我们可以在单个可交互地图上显示这些变化，活生生地展示它们发生的过程。这种做法保持了地图的直观性，同时又允许读者按自己的方法来探索数据。

几年前，我设计了一张地图，展示了沃尔玛在美国的发展史，如图8-34所示。动画始自于1962年阿肯色州罗杰斯市开办的第一家沃尔玛店铺，然后一直前进到2010年。每一家新店开张，地图上就会多出一个点。最初沃尔玛的增长比较缓慢，之后却开始像病毒一样在全美蔓延。它不断发展、成长，在各地进行大规模的收购后爆发。不知不觉中，沃尔玛已经无处不在了。

▶ 访问http://datafl.ws/197查看完整的沃尔玛店铺分布地图。

8

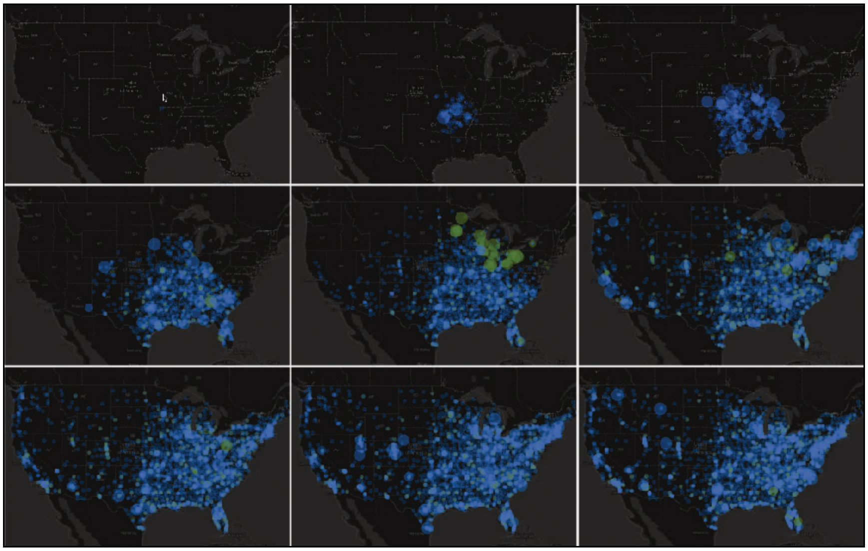


图8-34　表现沃尔玛增长的动画地图（另见彩插图8-34）

在当时，我做这张地图只是为了学习Flash和ActionScript而已，但它被四处转载，浏览量达到了数百万次。后来我又创建了一幅相似的地图，显示了塔吉特百货公司（Target）的发展历程（见图8-35），同样得到了广泛的传播。

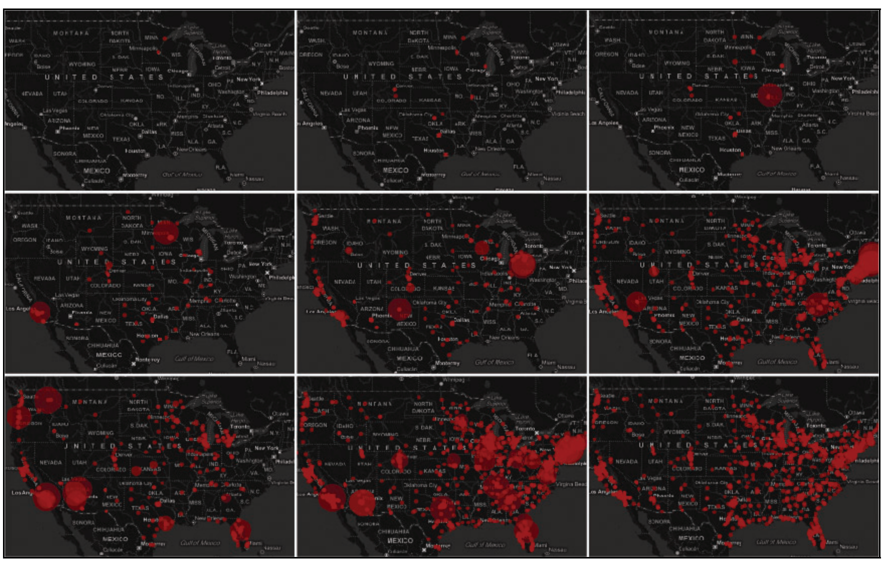


图8-35　表现塔吉特百货增长的动画地图（另见彩插图8-35）

▶ 访问http://datafl.ws/198观看塔吉特百货的发展历程。

人们对此如此感兴趣，有两个主要的原因。首先是因为动画地图能让他们看到在时间序列图中无法看到的变化模式。常规的图表只能显示每年新开张的店铺数量（如果你的故事只涉及这一层面，当然没问题），而动画地图则能更加“有机”地展现出增长的状态，对于沃尔玛这种级别的对象来说尤其如此。

第二个原因是这幅地图对普通大众来说非常容易理解。动画一开始我们就能明白自己在看什么。我并不是说那些需要花时间才能理解的可视化作品没有价值，实际情况往往是相反的。然而，网络读者对时间的忍耐力是有限的，而这幅地图的直观性（以及可以针对自己感兴趣的地点进行放大）自然会有助于人们分享的欲望。

创建动态增长地图

在本例中，我们用ActionScript来创建沃尔玛的增长地图。我们会用到ActionScript的地图库Modest Maps来生成可交互的基础地图。而其他代码则需要我们自己完成。大家可以在http://book.flowingdata.com/ch08/Openings_src.zip上下载完整的源代码。本节中我们不会详细讲解每一行代码和文件，而只会介绍最重要的部分。

▶ 访问http://modestmaps.com下载Modest Maps。

在第5章中，当我们用ActionScript和Flare可视化工具包创建堆叠面积图时，我曾强烈建议用Adobe的Flex Builder。它能让ActionScript的编写更为容易，同时让代码更有条理。当然，你也可以在一个标准文本编辑器里面写代码，但Flex Builder可以一站式解决你的编辑、调试和编译工作。本例假设你已经安装了Flex Builder，不过你要在Adobe网站上弄个ActionScript 3编译器下来自然也是没问题的。

说明　最近Adobe已经将Flex Builder改名为Flash Builder。两者间有一些小的变化，但不影响我们使用。

▶ 请下载完整的增长地图代码，以跟上本例的讲解。下载地址是http://book.flowingdata.com/ch08/Openings_src.zip。

首先打开Flex Builder 3，在左侧显示当前项目列表的边栏内单击右键，然后在弹出菜单中选择Import（导入），如图8-36所示。

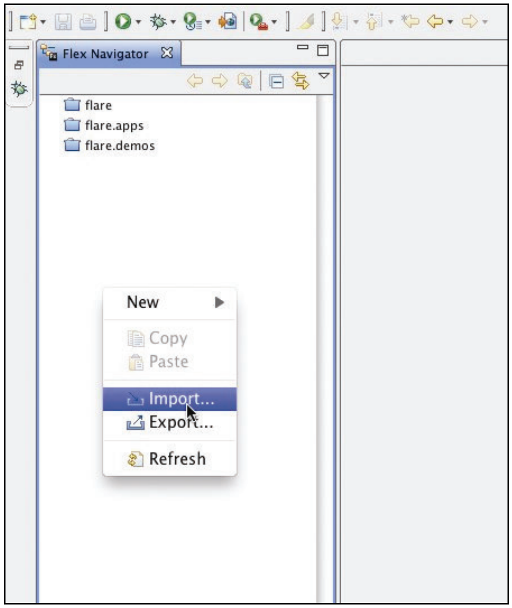


图8-36　导入ActionScript项目

选择Existing Projects into Workspace（现有项目到工作空间中），如图8-37所示。

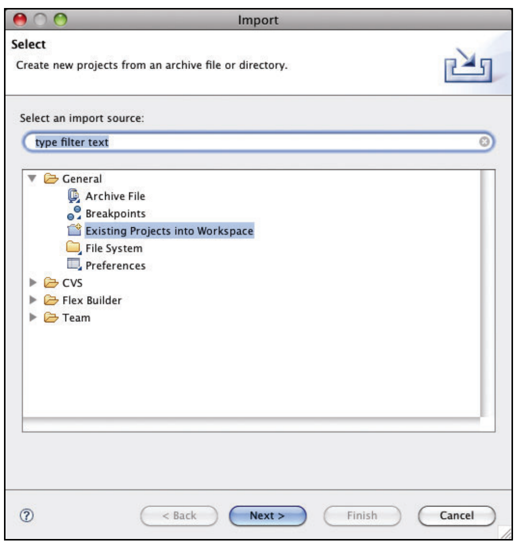


图8-37　现有项目

然后，如图8-38所示，单击Browse（浏览）按钮找到我们存放刚才下载代码的目录。在选择其根目录后，Openings项目会出现在项目窗口中。

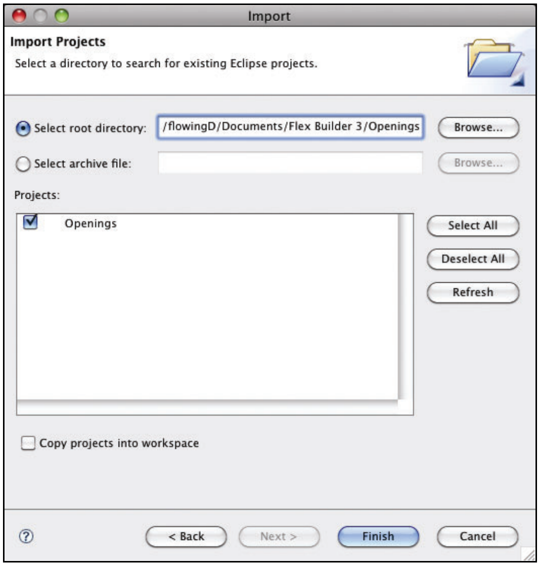


图8-38　导入Openings项目

我们在Flex Builder中的工作区界面应该类似于图8-39所示。

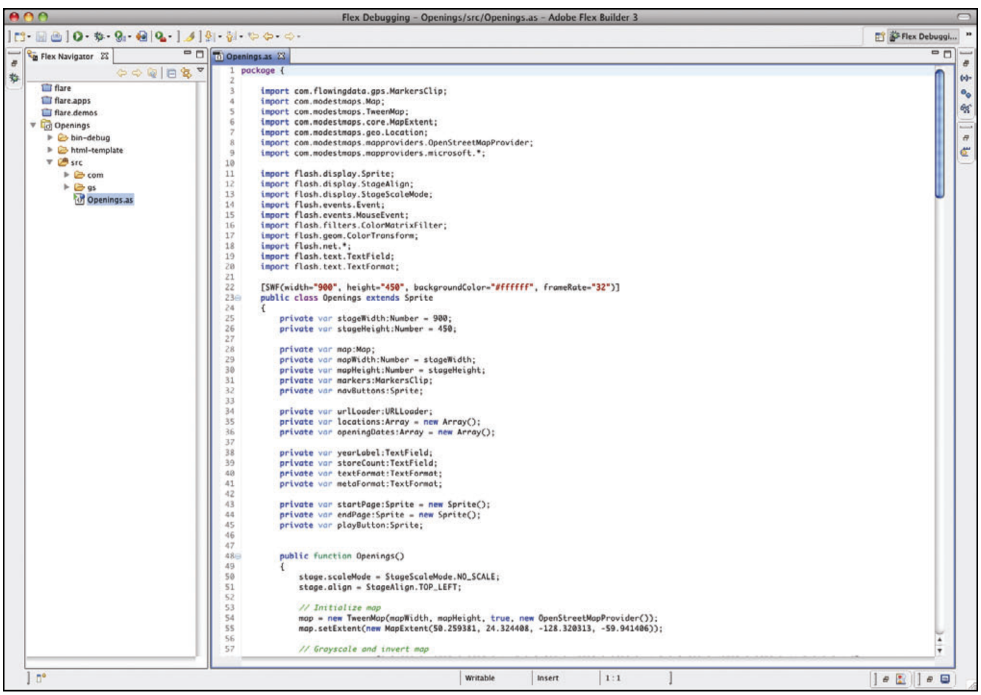


图8-39　导入项目后的工作区

所有的代码都在src文件夹中。这其中包含了com文件夹中的Modest Maps和gs文件夹中的TweenFilterLite，它们有助于实现过渡效果。

在导入Openings项目后，我们就可以开始构建地图了。这一过程分两部分，第一部分是创建可交互的基础地图，第二部分是添加标记符号。

● 添加可交互的基础地图

在Openings.as中，前面几行代码导入了所需要的工具包。

```
import com.modestmaps.Map;
import com.modestmaps.TweenMap;
import com.modestmaps.core.MapExtent;
import com.modestmaps.geo.Location;
import com.modestmaps.mapproviders.OpenStreetMapProvider;

import flash.display.Sprite;
import flash.display.StageAlign;
import flash.display.StageScaleMode;
import flash.events.Event;
import flash.events.MouseEvent;
import flash.filters.ColorMatrixFilter;
import flash.geom.ColorTransform;
import flash.text.TextField;
import flash.net.*;
```

第一段代码从Modest Maps工具包中导入了几个类，而第二段导入的是Flash提供的显示对象和事件类。每个类的名称是什么现在并不重要，在我们使用它们时自然会清楚。不过，第一段中的命名模式和其目录结构是相符的，以`com`开始，然后是`modestmaps`，最后以`Map`结尾。在你自己写ActionScript代码时，绝大多数时候也会这样来导入类。

在`public class Openings extends Sprite`之上，有关编译后的Flash文件的几个变量（宽度、高度、背景颜色和帧速率）都进行了初始化。

```
[SWF(width="900", height="450", backgroundColor="#ffffff",frameRate="32")]
```

然后，在类声明的后面，我们需要指定一些变量，并初始化一个地图对象。

```
private var stageWidth:Number = 900;
private var stageHeight:Number = 450;
private var map:Map;
private var mapWidth:Number = stageWidth;
private var mapHeight:Number = stageHeight;
```

在`Openings()`函数的括号里面，我们可以利用Modest Maps来创建自己的第一个可交互地图。

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.align = StageAlign.TOP_LEFT;

// Initialize map
map = new TweenMap(mapWidth, mapHeight, true, new OpenStreetMapProvider());
map.setExtent(new MapExtent(50.259381, 24.324408, -128.320313, -59.941406));
addChild(map);
```

和在Illustrator里面一样，我们可以把整个交互图表看作多个图层的累加。在ActionScript和Flash里面，第一层是舞台（stage，或者说画布）。我们将其设置为当放大地图时不要缩放对象，同时以舞台的左上方为标准来排放对象。之后我们用已经指定好的变量`mapWidth`和`mapHeight`对地图进行初始化，打开交互功能，并使用了来自OpenStreetMap的地图贴图。再之后我们设定了地图范围（`MapExtent`），将地图的框架限定在了美国。

`MapExtent()`函数中的坐标就是经纬度，它们可以控制边框以显示世界地图的哪个区域。第一个和第三个数字分别是左上角的经度和纬度，第二个和第四个数字分别是右下角的经度和纬度。

最后，通过`addChild()`函数将地图添加到舞台上。如果不对地图添加任何滤镜效果，编译代码后的地图就如图8-40所示。我们可以点Flex Builder左上角的Play（播放）按钮，也可以在主菜单里选择Run（运行）→Run Openings（运行Openings）。

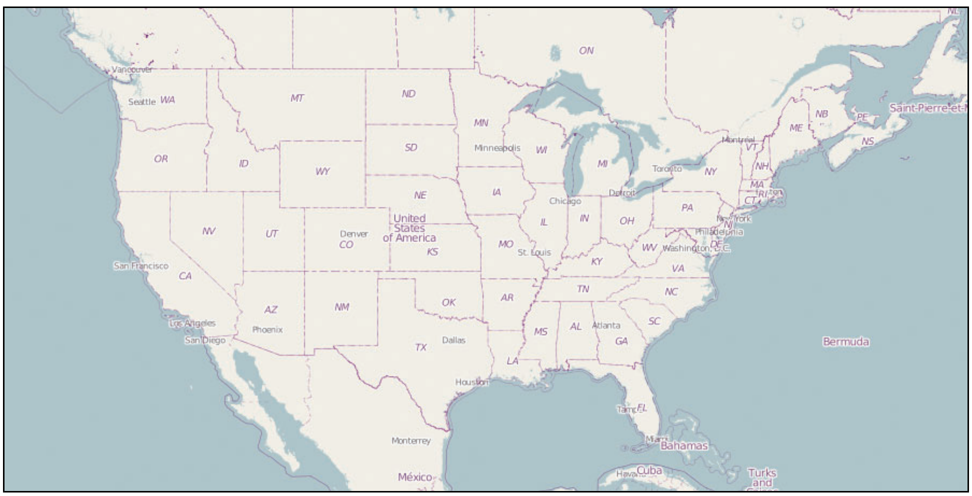


图8-40　使用了OpenStreetMap贴图的空白地图

运行Openings之后，结果会在系统默认的Web浏览器里打开。现在地图上还什么都没有，但我们可以随意拖动它，感觉还是很酷的。另外，如果你不太喜欢这种风格的地图贴图，也可以试试微软的道路地图（Microsoft road map，参见图8-41）或者雅虎的混合地图（Yahoo! hybrid map，参见图8-42）。



图8-41　使用了微软道路地图的空白地图

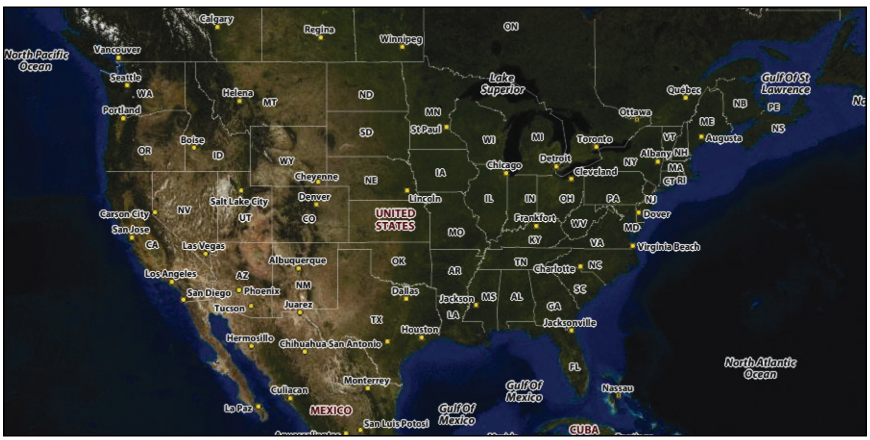


图8-42　使用了雅虎混合地图的空白地图

▶ 如果你愿意，也可以用你自己的贴图。在Modest Maps网站上有很棒的教程。

我们也可以应用滤镜来尝试修改地图的颜色。比如说，我们可以在刚才已写的代码下面添加以下地图，从而将地图改为灰度显示。其中的mat数组的长度为20，取值范围为0~1。每个数值代表各个像素点获得了多少红色、绿色、蓝色和透明度。

```
var mat:Array = [0.24688,0.48752,0.0656,0,44.7,0.24688,0.48752,
    0.0656,0,44.7,0.24688,0.48752,0.0656,0,44.7,0,0,0,1,0];
var colorMat:ColorMatrixFilter = new ColorMatrixFilter(mat);
map.grid.filters = [colorMat];
```

▶ 访问http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/filters/ColorMatrixFilter.html浏览Adobe的参考手册，以了解如何在ActionScript中用颜色矩阵来定制对象。

如图8-43所示，地图完全变灰了，这会为突出显示我们即将叠加在地图上的数据带来方便。地图的作用更像是背景，而不会喧宾夺主，转移读者的注意力。



图8-43　在应用滤镜之后得到的灰度地图

我们也可以通过颜色转换（color transform）来反转颜色。

```
map.grid.transform.colorTransform =
    new ColorTransform(-1,-1,-1,1,255,255,255,0);
```

这样会把白色变成黑色，黑色变成白色，如图8-44所示。



图8-44　通过颜色转换后黑白颠倒的地图

要想创建放大和缩小按钮，首先要写一个生成按钮的函数。你可能会认为应该会有一种默认的快速方法能实现这一点，但其实还是需要不少代码的。关于`makeButton()`的函数定义位于Openings类的底部。

```
public function makeButton(clip:Sprite, name:String, labelText:String,
action:Function):Sprite
{
    var button:Sprite = new Sprite();
    button.name = name;
    clip.addChild(button);

    var label:TextField = new TextField();
    label.name = 'label';
    label.selectable = false;
    label.textColor = 0xffffff;
    label.text = labelText;
    label.width = label.textWidth + 4;
    label.height = label.textHeight + 3;
    button.addChild(label);

    button.graphics.moveTo(0, 0);
    button.graphics.beginFill(0xFDBB30, 1);
    button.graphics.drawRect(0, 0, label.width, label.height);
    button.graphics.endFill();

    button.addEventListener(MouseEvent.CLICK, action);
    button.useHandCursor = true;
    button.mouseChildren = false;
    button.buttonMode = true;

    return button;
}
```

然后创建另一个函数来使用这个函数，并且绘制出我们想要的按钮。以下代码利用`makeButton()`函数创建了两个按钮，一个是放大，一个是缩小，并将它们安置在地图的左下角。

```
// 绘制导航按钮
private function drawNavigation():void
{
    // 导航按钮（放大和缩小）
    var buttons:Array = new Array();
    navButtons = new Sprite();
    addChild(navButtons);
    buttons.push(makeButton(navButtons, 'plus', '+', map.zoomIn));
    buttons.push(makeButton(navButtons, 'minus', '–', map.zoomOut));
```

```
    var nextX:Number = 0;
    for(var i:Number = 0; i < buttons.length; i++) {
        var currButton:Sprite = buttons[i];
        Sprite(buttons[i]).scaleX = 3;
        Sprite(buttons[i]).scaleY = 3;
        Sprite(buttons[i]).x = nextX;
        nextX += 3*Sprite(buttons[i]).getChildByName('label').width;
    }
    navButtons.x = 2; navButtons.y = map.height-navButtons.height-2;
}
```

不过，由于这是一个函数，如果我们不调用它，代码是不会执行的。在`Openings()`函数，也就是所谓的构造函数中，在滤镜代码的下面添加`drawNavigation()`。现在我们可以放大感兴趣的位置了，如图8-45所示。

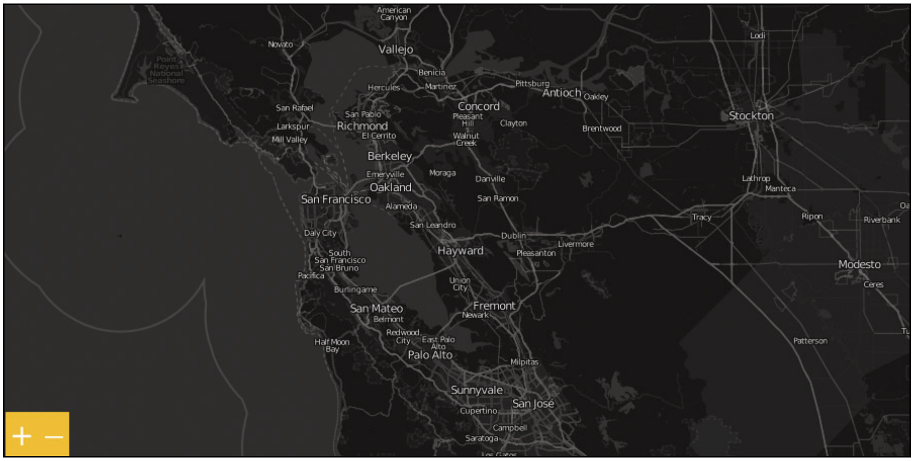


图8-45 能够放大缩小的地图

基础地图已经全部搞定了。我们选择了贴图，设置了变量，还让交互变得可行。

● 添加标记符号

下面的步骤是载入沃尔玛的店铺地址数据，并为每一家新开张的店铺创建标记符号。在构造函数里，以下代码从某个URL载入了一个XML文件。当文件完成下载后，就会调用一个名为`onLoadLocations()`的函数。

```
    var urlRequest:URLRequest =
        new URLRequest('http://projects.flowingdata.com/walmart/walmarts_
new.xml');
    urlLoader = new URLLoader();
    urlLoader.addEventListener(Event.COMPLETE, onLoadLocations);
    urlLoader.load(urlRequest);
```

8

很明显，下一步是创建这个onLoadLocations()函数。它会读取XML文件并将数据存储到一个数组中，方便后面使用。不过在此之前，我们需要在navButtons后面先对其他一些变量进行初始化。

```
private var urlLoader:URLLoader;
private var locations:Array = new Array();
private var openingDates:Array = new Array();
```

这些变量会用在onLoadLocations()里面。经度和纬度被存储在locations里面，而开张日期根据其具体年份被存储在openingDates里面。

```
private function onLoadLocations(e:Event):void {
    var xml:XML = new XML(e.target.data);
    for each(var w:* in xml.walmart) {
        locations.push(new Location(w.latitude, w.longitude));
        openingDates.push(String(w.opening_date));
    }
    markers = new MarkersClip(map, locations, openingDates);
    map.addChild(markers);
}
```

下一步是创建MarkersClip类。在之前讨论的那个目录结构中，在com目录下有一个名为flowingdata的目录，而在flowingdata目录下有一个gps目录。最终，在com → flowingdata → gps目录下是MarkersClip类。这就是即将存放所有沃尔玛标记符号（或者说在地图之上的数据图层）的容器。

和之前一样，我们需要导入将要用到的类。通常我们可以在需要的时候再在代码中添加它们，但为了简单起见，我们也可以一次性添加所有会用到的类。

```
import com.modestmaps.Map;
import com.modestmaps.events.MapEvent;

import flash.display.Sprite;
import flash.events.TimerEvent;
import flash.geom.Point;
import flash.utils.Timer;
```

前两个来自于Modest Maps，而后四个是Flash自有的类。然后我们在MarkersClip()函数之前设置好变量。同样地，我们可以在需要时才添加它们，但也可以现在就全部添加，从而直接前进到类里面的函数。

```
protected var map:Map;            // 底图
public var markers:Array;        // 存放标记符号的容器
public var isStationary:Boolean;

public var locations:Array;
```

```
private var openingDates:Array;

private var storesPerYear:Array = new Array();
private var spyIndex:Number = 0;  // 每年开张的店铺数索引
private var currentYearCount:Number = 0; // 目前已显示的店铺数
private var currentRate:Number;  // 待显示的店铺数
private var totalTime:Number = 90000;  // 约1.5分钟
private var timePerYear:Number;
public var currentYear:Number = 1962;  // 从初始年起

private var xpoints:Array = new Array(); // 转换过的经度
private var ypoints:Array = new Array(); // 转换过的纬度

public var markerIndex:Number = 0;
private var starting:Point;
private var pause:Boolean = false;
public var scaleZoom:Boolean = false;
```

在MarkersClip()构造函数中，我们存储好即将被传递到类的变量，并计算出诸如每年的时间和店铺的坐标值这些东西。你可以把这一步看作是设置。

变量storesPerYear存储的是在给定年份中有多少家新店开张。比如说，第一年有一家店铺开张，而第二年没有店铺开张。如果你要将这段代码用于自己的数据，就需要适当地更新storesPerYear这个变量。你也可以写一个函数来计算出每年开张的店铺数量或地址，以便提高代码的可重用性。为了简单起见，本例中直接将数组写死了。

```
this.map = map;

this.x = map.getWidth() / 2;
this.y = map.getHeight() / 2;

this.locations = locations;
setPoints();
setMarkers();

this.openingDates = openingDates;

var tempIndex:int = 0;

storesPerYear = [1,0,1,1,0,2,5,5,5,15,17,19,25,19,27,
    39,34,43,54,150,63,87,99,110,121,142,125,131,178,
    163,138,156,107,129,53,60,66,80,105,106,114,96,
    130,118,37];
timePerYear = totalTime / storesPerYear.length;
```

在MarkersClip类中还有其他两个函数：setPoints()和setMarkers()。前一个将经度和

纬度坐标转换成x轴和y轴坐标，而后一个则将标记符号放置在地图上（但并未显示）。以下是对`setPoints()`的定义。它使用了Modest Maps提供的一个内置函数来计算x值和y值，然后将新的坐标值存储在`xpoints`和`ypoints`中。

```
public function setPoints():void {
    if (locations == null) {
        return;
    }
    var p:Point;
    for (var i:int = 0; i < locations.length; i++) {
        p = map.locationPoint(locations[i], this);
        xpoints[i] = p.x;
        ypoints[i] = p.y;
    }
}
```

第二个函数`setMarkers()`使用了`setPoints()`存储的点坐标，然后将相应的标记符号放置在地图上。

```
protected function setMarkers():void
{
    markers = new Array();
    for (var i:int = 0; i < locations.length; i++)
    {
        var marker:Marker = new Marker();
        addChild(marker);
        marker.x = xpoints[i]; marker.y = ypoints[i];
        markers.push(marker);
    }
}
```

该函数还用到了一个定制的类`Marker`，如果下载了完整的源代码，我们就可以在com→flowingdata → gps → Marker.as中找到它。它基本上是一个容器，当你调用它的`play()`函数时，它就会被“点亮”。

现在我们有了位置数据，而且标记符号也载入到了地图上面。然而，如果现在编译代码并运行文件，我们依然只能看到一幅空白的地图。下一步是循环处理这些标记符号，让它们在正确的时间点亮。

`playNextStore()`函数调用了下一个标记符号的`play()`，然后为再下一个作好准备。`startAnimation()`和`onNextYear()`函数使用了计时器来递增显示每一个店铺。

```
private function playNextStore(e:TimerEvent):void
{
    Marker(markers[markerIndex]).play();
    markerIndex++;
}
```

现在再编译并运行动画，我们能够看到那些亮点，但它们并不配合地图一块进行缩放和平移，如图8-46所示。当我们左右拖动或者放大缩小地图时，那些绿色的气泡依然保持静止不动。

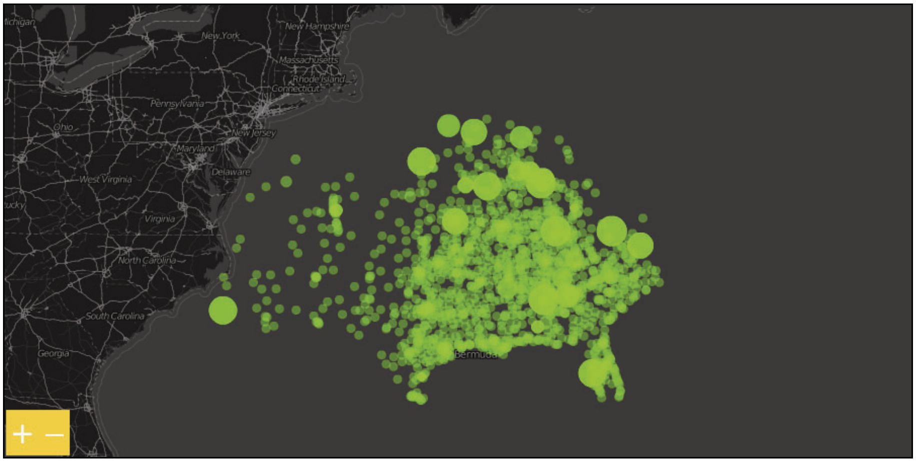


图8-46　不能缩放和平移亮点的增长地图

现在往构造函数里添加一些监听器，以便在地图移动时那些亮点也会跟着移动。无论何时，只要一个`MapEvent`被Modest Maps触发，一个在MarkersClip.as里定义的相应函数就会被调用。比如下面的第一行代码，只要用户单击地图的缩放按钮，`onMapStartZooming()`就会被调用。

```
this.map.addEventListener(MapEvent.START_ZOOMING,
  onMapStartZooming);
this.map.addEventListener(MapEvent.STOP_ZOOMING,
  onMapStopZooming);
this.map.addEventListener(MapEvent.ZOOMED_BY, onMapZoomedBy);
this.map.addEventListener(MapEvent.START_PANNING,
  onMapStartPanning);
this.map.addEventListener(MapEvent.STOP_PANNING,
  onMapStopPanning);
this.map.addEventListener(MapEvent.PANNED, onMapPanned);
```

这样我们就得到了最终的地图，如图8-47所示。

沃尔玛开店的故事显示了一种有机的增长。公司从一个地方开始，缓慢地向外扩张。很明显，并不是所有连锁店都是如此。比如，塔吉特的发展看上去就没有这么有计划性。好市多的发展不像前两者那样充满戏剧性，因为它扩张的区域较少，但其发展战略似乎是先在沿海地区站稳脚跟，然后向内地进军。

不管怎样，这是一种非常有趣的浏览数据方式。不停增长的地图似乎能激发人们的想象力，而且他们会想知道其他连锁公司（例如麦当劳或星巴克）的扩张情况。现在我们已经有了代码，实现这一点应该会容易得多。困难的部分只在于如何找到数据了。

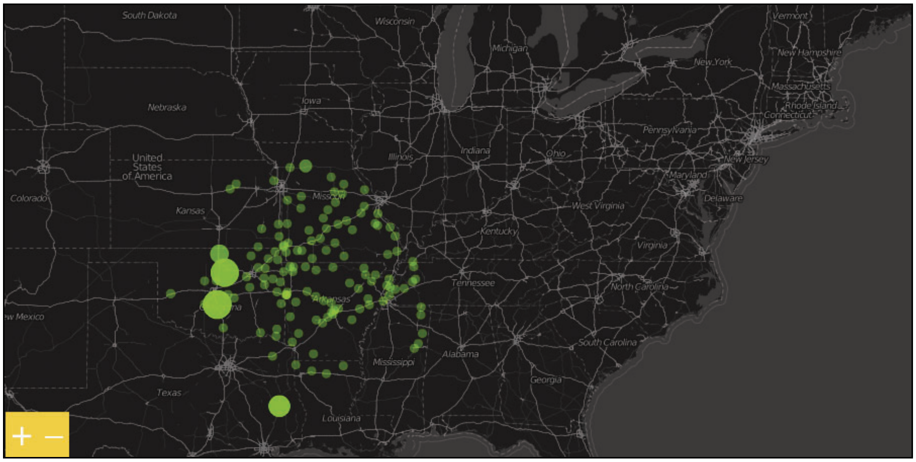


图8-47　显示沃尔玛增长情况的完整可交互地图

8.5　小结

地图是一种有些棘手的可视化类型，因为除了手中的数据之外，我们还必须处理地理维度。但是，由于它们的直观性，地图也能为我们带来丰厚的回报：一方面丰富了我们呈现数据的形式，另一方面它也比静态图表更加利于我们深入地探索数据。

正如大家在本章这些例子中所看到的，在处理空间数据时有很多的可能性。只需要一些基本的技巧，我们就可以可视化众多的数据集，讲出各种有趣的故事。以上只是冰山一角而已。我的意思是，人们需要上大学、读研读博才能获得制图学和地理学的学位，所以除了以上我们讲的那些之外，肯定还有很多可以发挥的地方。你可以试试分区统计图（cartogram），它可以根据某种度量来确定地理区域的大小；也可以在Flash里面添加更多的交互行为；或者将地图与图表混合使用，为数据表现出更多的细节和探索方向。

在线地图已经变得非常普遍，而且它们受欢迎的程度会随着浏览器和工具的发展而继续增长。在我们的沃尔玛地图实例中用到的是ActionScript和Flash，但也可以通过JavaScript来实现。选择何种工具仅仅取决于我们自身的意图，如果对工具没有要求，那就选择你最得心应手的那一个。其实，关键在于逻辑而非软件。语法可以改变，但表现数据的手法是一致的，我们讲故事的起承转合也并无不同。

9 有目的地设计

当我们探索自己的数据时，没必要关心太多讲故事的条条框框，毕竟我们自己就是讲故事的人。不过，在使用图表来展示信息时，不管对象是一个人、数千人还是上百万人，单单一幅图表很可能是不够的。

毫无疑问，我们希望其他人能理解我们得出的结果，说不定还能从中发展出他们自己的故事。但是，读者在刚刚拿到数据时很难知道应该关心哪些问题。降低门槛是我们的责任和义务。我们如何设计图表将直接影响到读者如何理解数据。

9.1　让自己作好准备

我们只有充分了解了自己掌握的原始材料，才有可能用数据把故事讲好。而这往往是设计数据图表时最容易被忽略的。在开始阶段，的确很容易陷入对最终效果的憧憬和痴迷中。你想要一张看起来精彩、漂亮又有趣的图表，这没什么问题，但如果还不知道要可视化什么东西，谈这些无疑是痴人说梦。最后你可能只能得到类似图9-1这样的东西。你连数据集中有什么数据都不清楚，又怎样去突出那些亮点呢？

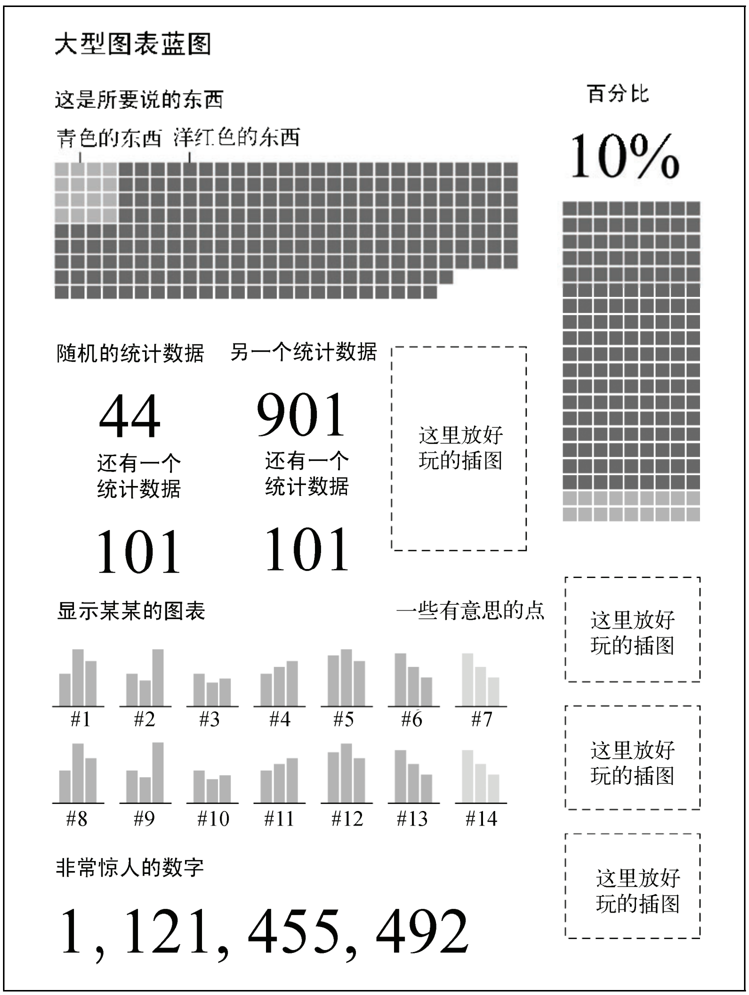


图9-1　大型图表蓝图。大而空洞

要研究数字和其中的度量标准。找出它们来自哪里，又是如何被估算的，还要检查它们是否合乎常理。正是这些早期的数据收集过程让《纽约时报》的图表变得如此优秀。我们看到的只是

报纸或者网站上的最终结果，却不会感受到在动笔绘制之前所需要付出的大量努力。很多时候，仅仅只是将所有数据排好序所花的时间就要远远超过绘制图表的时间。

所以，当下次拿到一个数据集的时候，请试着不要立刻就埋头开始设计。那是懒人的办法，而且会在最后暴露出所有的缺陷。最好还是花点时间了解你的数据，研究这些数字的上下文背景。

提示　可视化的目的就是传播数据，所以要花时间了解什么才是你的图表的基础，否则你最终只是在堆砌数字而已。

把数字丢到R里面检验，阅读附带的说明资料，以确保你了解每一种度量指标所代表的意义，并且检查是否有看上去奇怪的地方。如果有，而你又不知道背后的原因，那么就去联系数据的提供者。人们通常会很高兴有人用到了他们发布的数据，也会很希望能找出其中的错误并加以修改。

在充分了解了数据之后，就可以开始设计图形了。我们可以打个比方。还记得电影《龙威小子》[①]中丹尼尔刚开始学习武术的那一段吗？宫城先生让他给一堆汽车打蜡、给地板抛光、整修花园的栅栏，这让丹尼尔非常沮丧，因为他觉得做这些事情毫无意义。理所当然，最后各种阻挡和出拳的技巧一下子全都自然地涌现、达到了心神合一，因为他其实一直都在练习正确的功夫动作。对于数据来说道理也是一样的。尽量多了解数据，最后用视觉手段来讲故事自然就会水到渠成。如果你还没看过这部佳作，赶紧承认吧，然后把《龙威小子》加到你的Netflix[②]列表里面去。

9.2　让读者作好准备

作为数据图表设计师，我们的任务是将自己所知道的内容传达给受众。受众一般都没有看过原始数据，所以如果没有任何的解释或提前说明，他们看到的东西很可能和我们看到的完全不同。对此我的经验就是假设所有人在看到我的图表之前都是盲目的。实际上，通过Facebook、Twitter以及其他博客链接等分享方式，我发现我的假设并不算特别离谱。

例如，图9-2是我制作的一幅动画地图的截图。如果你在此之前没看过这张图，可能会毫无头绪。而根据我们在第8章中讲过的实例，你最好的猜测可能是某家连锁店的扩张情况。

▶ 访问http://datafl.ws/19n观看完整的地图动画。

实际上，这幅地图显示的是巴拉克・奥巴马总统就职典礼期间全世界发布的带地理标记的Twitter微博，时间是美国东部标准时间2009年1月20日，星期二中午。动画从星期一早上开始，随着时间的推进，越来越多人起床，并以稳定的速率发推。随着就职典礼的临近，每小时发推的

---

①《龙威小子》是1984年出品的一部美国功夫电影。其侧重点不在于武术，而是一个少年的成长过程，在上映后受到了青少年的热烈欢迎，并被评为1984年的最佳影片之一。2010年出了重拍版，其中成龙也有出演。

② 美国一家提供电影、视频流媒体服务和DVD租赁业务的公司。——编者注

数量开始增长，而欧洲人则在美国人开始入睡时开始有所行动。然后星期二的早晨来临了，随着就职典礼的正式开始Twitter突然爆发，群情激昂显而易见。大家可以在图9-3中看到这一过程。如果我在图9-2中加入这一上下文背景，可能就会让人容易理解得多。

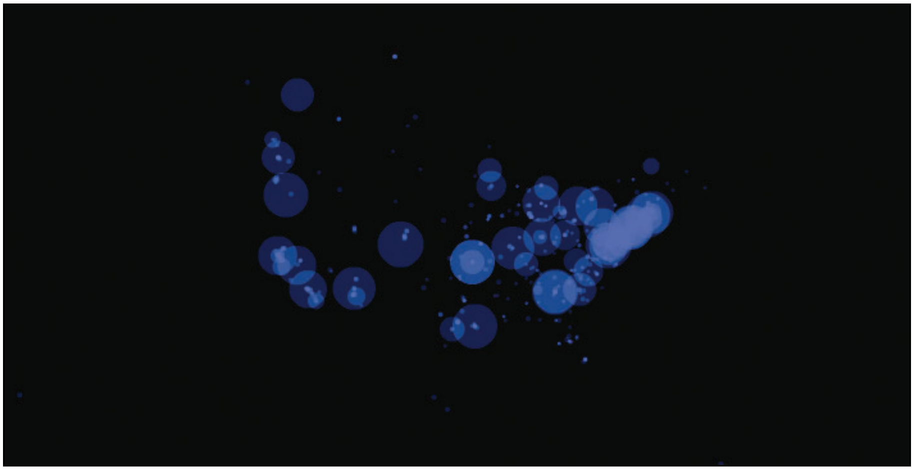


图9-2　没有标题和上下文背景的地图

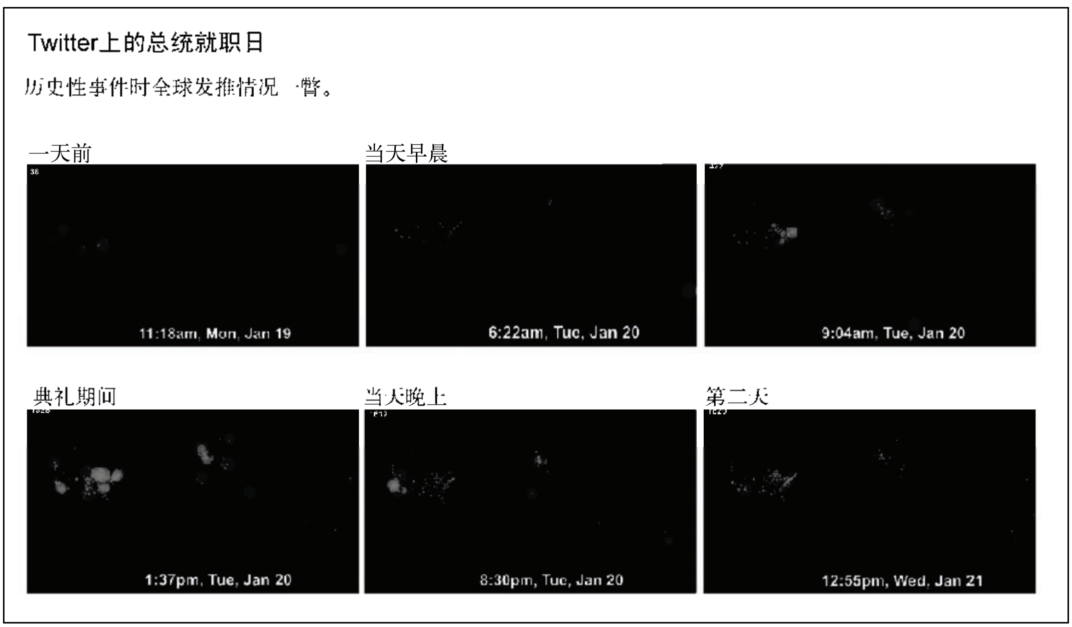


图9-3　巴拉克・奥巴马总统就职典礼期间的Twitter微博（另见彩插图9-3）

我们不必为每幅图表都配上一大段文字，但添加一个标题和一小段引文加以解释总是会很有帮助的。如果在图表中的某处再提供一个链接可能会更好，这样一来即使图表被分享到其他网站上，人们依然可以找到原始出处，看到你随图发表的文章。否则，很快就会变成电话传话游戏，在你意识到以前，你精心设计的图表就会完全被人曲解了。互联网就是这么诡异。

再举一个例子，图9-4中的图表是一个简单的时间轴，显示了当代十大数据泄漏事故。

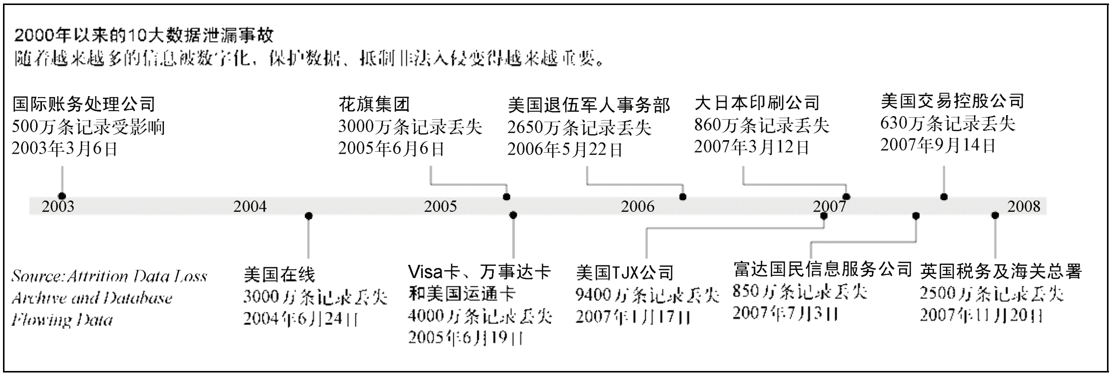


图9-4　2000年以来的主要数据泄漏事故

这张图很基础，只有10个数据点，不过在发表到FlowingData上时，我提到从2000年到2008年间数据泄漏事故发生的频率越来越高。这幅图表最后也被相当多的人分享，甚至其变体还登上了《福布斯》杂志，而几乎所有分享的人都提到了我说的频率问题。但我觉得，如果我之前没有提到这一点的话，并不会有那么多人对这幅图表如此上心。

这件事给我们上的一课是：不要假设读者知道一切，或者能从你的图表中发现所有细节。这对于互联网用户来说尤其正确，因为网友们已经习惯于点击前往下一处了。

但这并不是说人们不会花时间去观察数据。可能有人看到过，OkCupid的官方博客里经常有一些比较长的博文，根据网站在线交友的数据进行分析，然后将结果表现出来。这些文章的题目包括“第一次约会时提什么问题比较好”和“美人与数学”等。

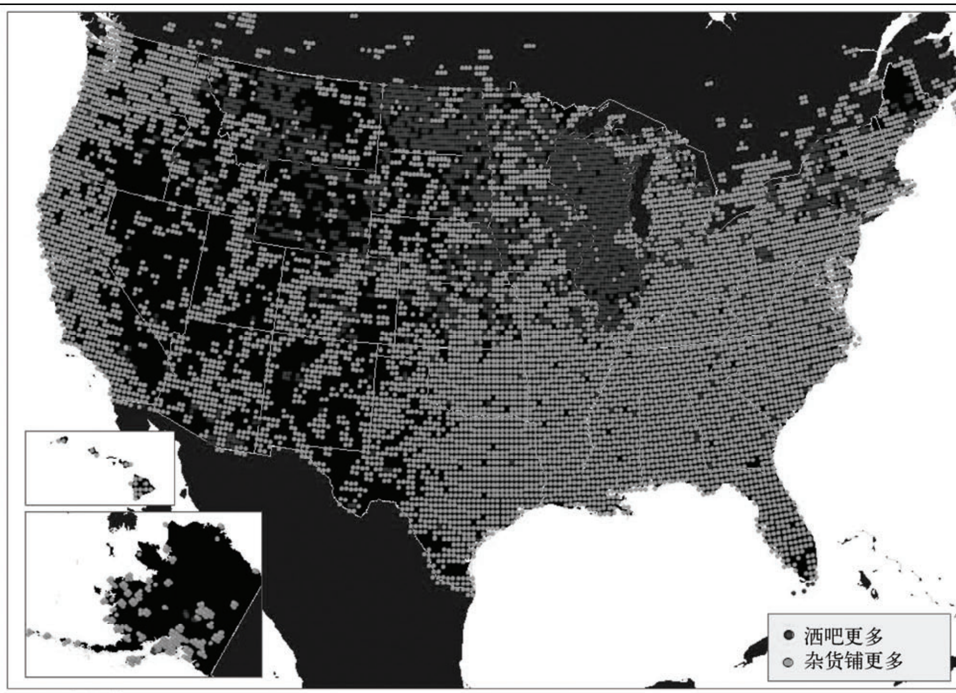
该博客上的文章被浏览过上百万次，人们非常喜欢看OkCupid的那些家伙们所说的话。除了文章里面大量的上下文背景之外，来阅读博客的人们也有各自的上下文背景。因为这是关于约会、异性方面的数据和发现，人们总是能将它们和自身的经验联系起来。例如，图9-5显示了亚裔男人通常喜欢什么。这幅图表来自于OkCupid一篇关于人们喜欢什么的文章，其中根据人种和性别进行了分类。嘿，我自己就是亚裔，而且我是男的。你看，很快就产生了联系，不是吗？

而另一方面，要是你的图表主题是污染情况或全球债务，如果没有解释清楚，那么大部分读者可能很难买账。

还有些时候，不管我们怎样说明，人们就是不喜欢在网上阅读，总是囫囵吞枣。有一次，我在FloatingSheep上发表了一张地图，对美国的杂货铺数量和酒吧数量进行了对比，如图9-6所示。红色表示该地区的酒吧比杂货铺多，桔色表示相反。FloatingSheep上的家伙们称之为“美国的啤酒肚”。



图9-5　亚裔男人都喜欢什么，来自于OkCupid在线交友资料



酒吧和杂货铺：不同的颜色代表在Google地图上的某个具体位置搜索某一关键词得到的结果数量更多。例如，红色圆圈表示在当地搜索“酒吧”得到的结果数量超过了“杂货铺”的结果。数据收集于2008年8月。

图9-6　美国哪些地方的酒吧比杂货铺数量多（另见彩插图9-6）

在发表文章的时候，我对该地图的准确性存有疑问，所以在结尾我是这么写的：“有没有住在当地的朋友还足够清醒？帮我确认一下你们那里是不是这样吧。我猜你们的评论里错别字肯定不少，而且话都说不清，甚至满口酒气呢。”那么我得到了什么教训呢？在网上，冷幽默和挖苦往往很难得到正确解读，更何况大部分人根本就懒得通读你写的文章。实际上我根本不认为自己会看到满口酒气的评论。很多人知道我是在开玩笑，但还是有不少人感到受了冒犯。正如我所说的，互联网就是这么有趣。

9.3 视觉提示

在第1章中，我们看到了“编码”是如何生效的。简单来说，我们手上有数据，而这些数据被根据几何、色彩或动画原理进行了编码。然后读者对这些形状、颜色和运动进行解码，将它们重新映射为数字。这就是可视化的基础。编码是一种视觉上的翻译过程。而解码则能帮助我们从不同的视角观察数据，并且从中找出变化的模式。而这种模式用表格是无法体现的。

这种编码一般都比较简单易懂，因为它们遵循的是数学规则。较长的柱形代表较高的数值，面积较小的圆形代表较低的数值，等等。虽然在这一过程中会由计算机作出很多决定，但是针对手中的数据集选择何种合适的编码方式，这一自主权依然掌握在我们自己手上。

通过前面章节中的所有实例，我们发现好的设计不仅体现出了美学的价值，而且还能让图表更加易于阅读，甚至还能影响读者对数据或故事产生的感觉。由R或Excel默认产生的图表看上去非常原始和机械，但这也不一定是坏事。也许对于学术报告来说这就已经足够了。或者说，如果文章主体更加重要，而图表只是对其的一种补充，那么最好不要让图表喧宾夺主，分散读者的注意力。图9-7就显示了一种极为朴素的柱形图。

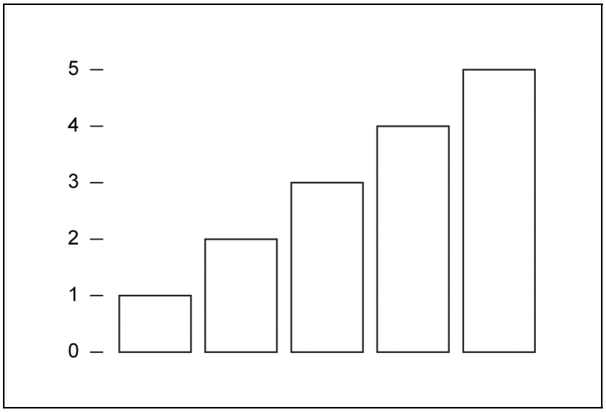


图9-7 朴素的柱形图

但是，如果你希望让图表更加吸引眼球，简单地改一下颜色就能收到不错的效果。在图9-7的基础上仅仅调整了背景色和前景色就得到了图9-8。

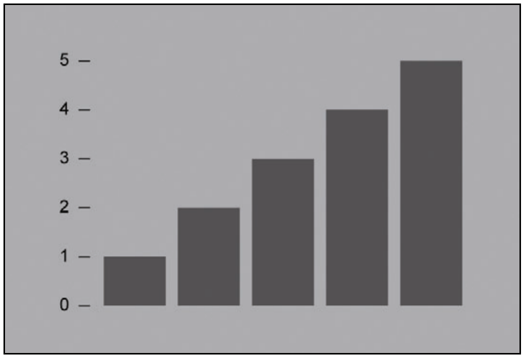


图9-8　暗色主题的默认图表

较暗的配色方案可能适用于较严肃的主题，而明亮的配色方案（如柠檬黄）则能让人感到更加愉快、无忧无虑（见图9-9）。

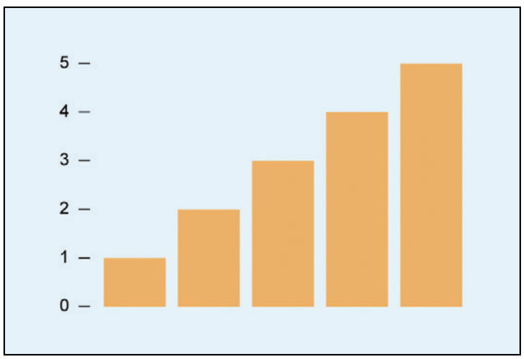


图9-9　亮色主题的默认图表

当然，我们并不总是需要鲜明的配色方案。如果愿意，你大可以使用中性的配色，如图9-10所示。

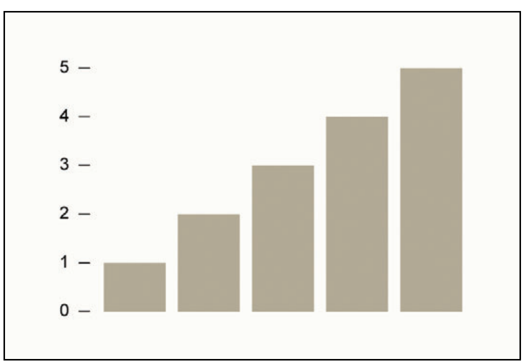


图9-10　中性配色的默认图表

关键是，颜色的选择在数据图表中可以扮演重要的角色。它可以调动（或者不调动）读者的情绪，还有助于提供上下文背景。选择合适的颜色来传达准确的信息是我们的责任，色彩应该与所讲的故事相搭配。就像在图9-11中显示的，简单的颜色变化可能会完全改变数据的含义。这幅图表来自于设计师David McCandless和二人设计组合Always With Honor，它探索了不同文化中色彩的意义。比如说，黑色和白色在大部分文化中都用于表现死亡，而在南美文化中，则通常用绿色来表现这一主题。

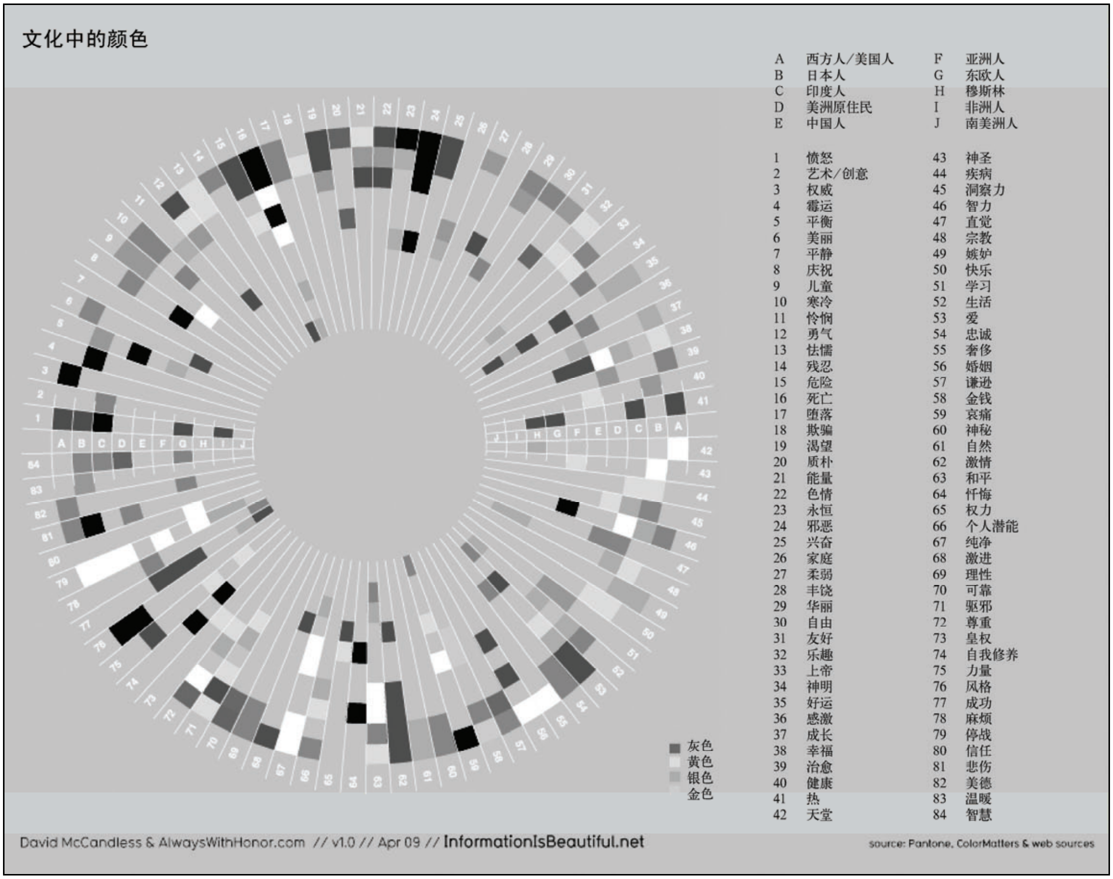


图9-11　由David McCandless和Always With Honor设计的“文化中的颜色”（另见彩插图9-11）

与颜色相似，我们也可以通过修改几何形状来得到不同的外观、感觉和含义。例如，图9-12显示了一个随机生成的堆叠柱形图，使用的工具是可视化学者Mike Bostock开发的Data-Driven Documents[①]。图形有平直的边缘、明显的数据点，以及加上波峰和波谷。

① Data-Driven Documents又称为“D3”，它是一个JavaScript库，通过HTML、SVG和CSS来绘制图表。网站为http://d3js.org。

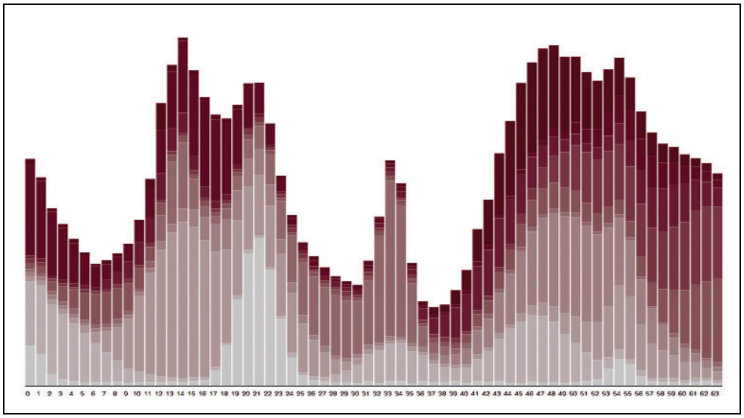


图9-12　随机生成的堆叠柱形图

如果用流线图（streamgraph）来显示相似的数据，我们就会得到完全不同的感觉，如图9-13所示。图形更加连贯而律动，而且不再有波峰和波谷，取而代之的是收缩和膨胀。尽管如此，这两种图表所用的几何原理是相似的。流线图可以看作是更加平滑的堆叠柱形图，其水平轴在中间而不是位于底部。

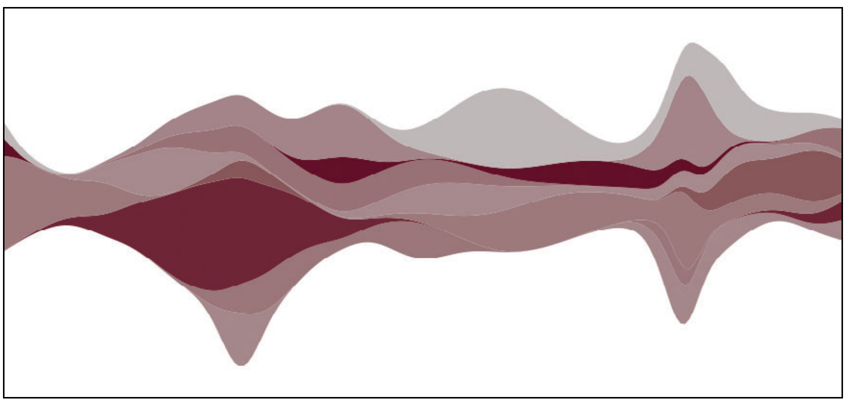


图9-13　随机生成的流线图

▶ 请查阅Lee Byron和Martin Wattenberg的论文“Stacked Graphs—Geometry and Aesthetics”（堆叠图表——几何与美学），了解更多流线图的信息。有一些工具包可以让我们自己设计流线图，例如Protovis和D3。

有时候在我们组织形状和颜色的过程中，上下文背景自然而然就会显现出来。图9-14是一幅我创作来庆祝节日的图表。上半部分显示了浸泡火鸡时需要的原材料，下半部分则是在送入烤箱

前需要填入火鸡肚子的东西。

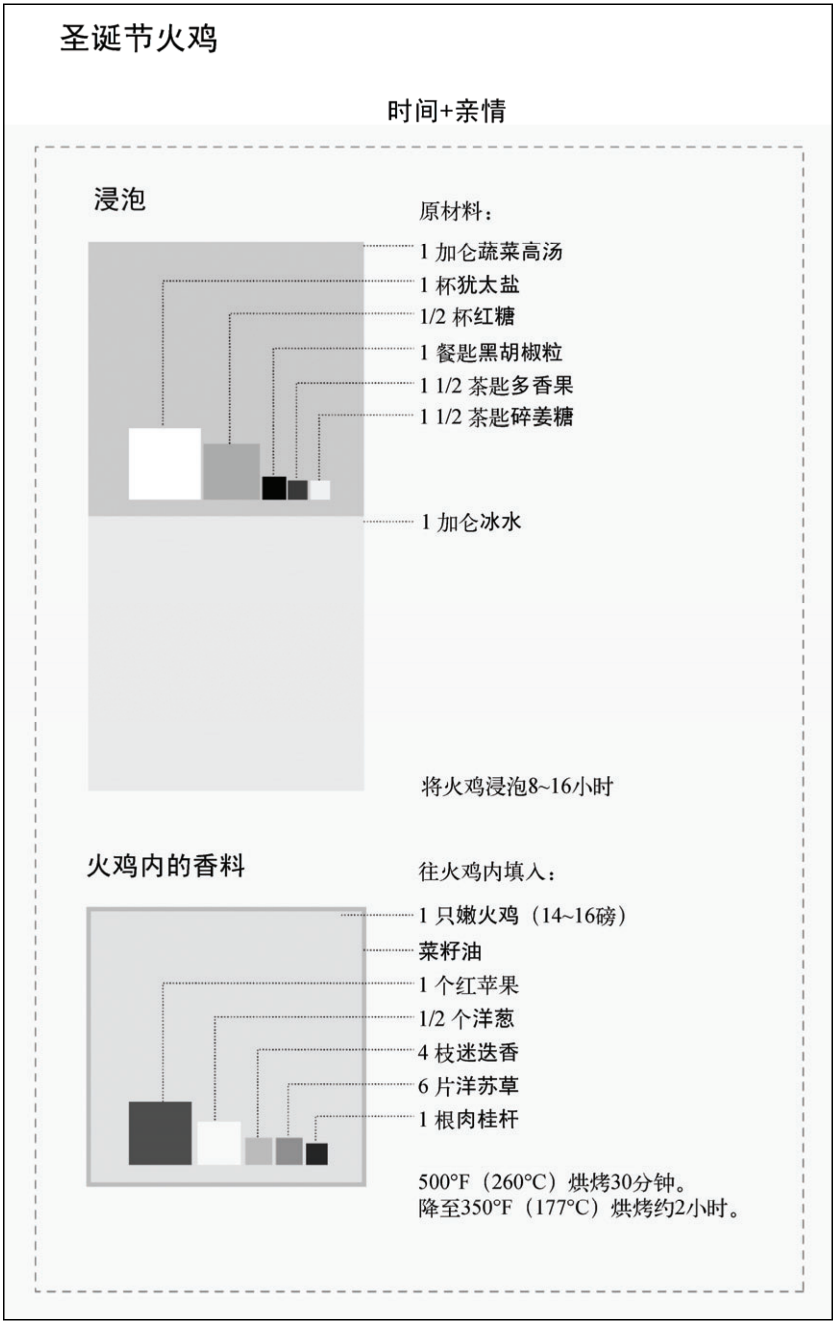


图9-14 圣诞节火鸡菜谱

归根结底，从最本质的层面来说，可视化就是将数据（可以是数字、文本、类别或任何其他东西）转换为视觉元素。有一些视觉元素比其他元素更具提示性，但不一定适用于所有数据集。对某种数据集完全不适用的方法，对另一种数据集来说可能是黄金搭配。通过不断的实践，我们很快就能找到最适合当前设计意图的可视化方法。

9.4　好的可视化

尽管数个世纪以来人们一直在绘制图纸和图表，但只在过去几十年里研究人员才开始分析哪些图表有效、哪些图表无效。从这个角度来说，可视化仍然属于一个崭新的领域。对于可视化究竟是什么，业界目前仍未达成共识。可视化是否等同于计算机根据一些规则生成的结果？如果有人在设计过程中进行了人工干预，是否就表示它不再是可视化作品？信息图到底属于可视化范畴还是应该有自己的专有分类？

在网上我们可以找到许多有关信息图和可视化之间异同的讨论，以及各种各样试图定义可视化的论述。这些讨论往往流于无休止的来回反复，却得不出任何结果。这些互左的意见导致对数据图表的优劣评判标准也参差不齐。

比如说，统计员和分析员普遍认为可视化就是他们在分析中用到的传统静态图表。如果该图表或者交互作品无助于分析工作，那么就毫无价值，是失败的。而另一方面，如果和一位图形设计师讨论同一幅图表，他们可能会认为它是成功的，因为它突出了令人感兴趣的数据，而且表现的方式非常迷人。

我们所需要做的就是把这些观点整合到一起，至少也应该尽量综合考虑。分析型头脑的人可以从设计师那里学到如何让数据更具有叙述性、更便于人们理解，而设计型头脑的人可以学到如何像分析员那样更深入地挖掘数据。

我不会去定义可视化是什么，因为这种定义不会影响我的工作方式。我会考虑我的受众群体以及面前的数据，然后自问最终的设计是否能达到要求。如果能，那很好；如果不能，我就会回头重新画草图，思考如何才能进一步优化，让图表回答出我对数据产生的那些问题。总而言之，重点在于你设计图表的目的是什么，你想讲什么样的故事，以及你打算跟谁讲。将上述所有内容都考虑进来，你就能成功。

9.5　小结

很多与数据打交道的人认为设计只是一种让图表更加好看的手段。这话并不错，但设计还能够让我们的图表更加易于阅读、理解和使用。与默认生成的原始图表相比，经过设计的图表可以帮助人们更好地理解数据。我们可以让图表不再杂乱，突出数据中的重要内容，甚至还可以唤起读者的情感共鸣。数据图表也可以娱乐大众，既丰富多彩又包罗万象。有时候它可能就只是为了娱乐而已，这取决于你自身的意图，但不管是何种作品（可视化也好，信息图也好，数据艺术品也好），还是得让数据来引导你的设计。

如果你手上的数据集非常庞大，而且你不知道如何着手，那么最好就是以一个问题开始。你希望从中了解什么？你是否在找周期性的模式？或者多个变量之间的联系？异常值？空间关系？然后再回头观察数据，看看是否能回答刚才的问题。如果没有你需要的数据，那么就去再找一些来。

在有了数据之后，你就可以运用从本书中学到的技巧来讲有趣的故事了。不过，这还不是终点。将你现在的所有设计产出看作是基础，然后再度挖掘。所有我们喜欢的数据图表的核心都是一样的：某个数据类型，再加上某种可视化方法（你现在已经知道这些方法的原理了）。你可以根据这些已有的素材尝试更高级、更复杂的图表。加入交互行为，混合多种图表，或者补充一些照片和文字以提供更多的上下文背景。

请记住，数据只是对真实生活的一种表现。当我们在可视化数据时，其实是在可视化我们身边以及这个世界上正在发生些什么。我们的观察范围可以从微小的个体到广袤的宇宙。仔细研究数据，我们就能讲出大多数人从不知晓、但却渴望听到的故事。如今我们这个世界上的数据比以往任何时候都要丰富，而且人们希望了解它们背后蕴涵的信息。现在你可以告诉他们了。祝你玩得开心。

延伸阅读

（即将出版）

看得见的信息

[日] 木村 博之 著

信息图学的力量：巧用图形与人沟通

[加] Mark Smiciklas 著

信息架构用户体验设计

[日] 長谷川 敦士 著